OMNIBUS User's Manual

The OMNIBUS User's Manual was prepared by the technical staff of Innovative Integration on October 23, 2001.

For further assistance contact:

Innovative Integration
2655 Park Center Drive
Simi Valley, California 93065

PH: (805) 520-3300
FAX: (805) 579-1730
email: techsprt@innovative-dsp.com
Website: www.innovative-dsp.com

VSS\Omnibus Modules\Documents\Omnibus Manual\OMNIBUS.book

List of Figures

List of Tables

CHAPTER 1

OMNIBUS Description and Specification

The OMNIBUS standards provide a fast, flexible, 32-bit wide mezzanine I/O expansion capability for Innovative Integration's DSP and data acquisition boards. OMNIBUS compatible hosts (including the M44, SBC6x, M6x, cM6x, Chico, cChico, ChicoPlus, cChicoPlus, Hombre, and cHombre) can be equipped with modules supporting a wide range of I/O specifications and signal standards. This permits access to real world hardware for data acquisition, control systems, and communications applications. Because the OMNIBUS standard is modular, users can customize any of the available OMNIBUS hosts with application specific interfaces for zero NRE cost. The OMNIBUS Modules bring do-it-yourself, cost effective DSP hardware to the OEM user.

This manual details the OMNIBUS standards, including signal definitions, mechanical specifications, host processor board addressing, software development, library support, and compatibility issues. Information on designing custom OMNIBUS module hardware is also included for those users who have specific interfacing needs not covered by the existing line of OMNIBUS modules. Finally, complete documentation is provided for Innovative Integration's lineup of OMNIBUS modules. Each module is described in separate chapters, with information regarding register access and high level C library support for the particular devices on each module along with hardware specifications.

Please refer to the Omnibus.hlp and/or Zuma.hlpWindows help file for a detailed description of each of the C functions referred to within this manual. This help file is usually installed on your PC in the C:\<target boad>\documentation directory during the Zuma Toolset installation process.

OMNIBUS Introduction

The OMNIBUS is a modular mezzanine bus standard that allows a host processor board to accept plug-in modules, which in turn implement hardware interfaces to external equipment. Modules may be memory mapped into the host boards memory, while certain host cards provide additional alternative access methods (see below for details).

Mechanically, OMNIBUS is implemented as a 4.6” by 2.0” mezzanine board, which uses two 50 pin high density connectors (called the “bus connectors”) to implement the bus connections to the host hardware. An additional connector (called the “I/O connector”) for use as a pass-through to connect to the external hardware. The bus connector pinout is standard and allows the modules access to the host data bus, address bus, bus control lines (such as clock and wait state control), timebase signals, processor interrupts, and power. The I/O connector pinout is unspecified and all fifty pins are available for use in connecting to external equipment. The block diagram below describes the connector arrangement.

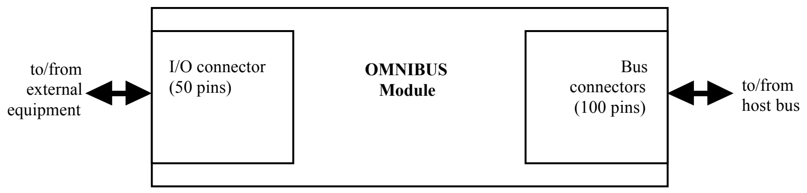


FIGURE 1. **OMNIBUS Module Connector Block Diagram**

Hosts can provide as many OMNIBUS slots as is mechanically and electrically possible given the host card’s physical size, memory map, and power supply capabilities. Most Innovative hosts provide two slots (M44, M6x, SBC6x, ChicoPlus, and Hombre) while others provide three slots (cM6x). The Chico card provides a single slot.

Bus Connectors

The OMNIBUS bus connectors provide access to the hosts processor’s bus for the purpose of communication with the host. Present on the bus connectors are the data, address, bus control, timing, and power signals necessary to interact with the host processor. The two 50 pin bus connectors are nominally named “Bus Connector A” and “Bus Connector B” and each carries a subset of the signals.

The following tables give the pinouts for the bus connectors.

Pin Number	Pin Name	Function	Direction (from host)
1, 19	DVCC	Digital +5V	O, power
2, 20	DGND	Digital ground	O, power
3-18	D0-D15	Data bus 0-15	I/O
21, 43, 40, 45, 39, 26, 27	A0-A6	Address bus 0-6	O
28	RST*	Reset (active low)	O
29	EINTx*	External interrupt	I
30	RDY*	Bus ready (active low)	I (open-collector)
31	CLK	Clock	O
32	TMRx	Timer channel	O
33	R/W*	Read/write	O
34	DDS	9850 timebase	O
35-38	IOMODx*	OMNIBUS decoded selects (active low)	O
25	-12V	-12V	O, power
23	+12V	+12V	O, power
41, 42	AGND	Analog ground	O, power
22, 24	-15V	Analog –15V	O, power
44, 46	+15V	Analog +15V	O, power
47, 49	+5V	Analog +5V	O, power
48, 50	-5V	Analog -5V	O, power

TABLE 1. **Bus Connector A Pinout**

Pin Number	Pin Name	Function	Direction (from host)
1, 3-6	A7-A11	Address bus 7-11	O
2, 19, 20, 49, 50	DGND	Digital ground	O, power
7-18	-	Reserved	NA
21	TMRx	Timer channel	O
22	EXT_TRIGx*	External trigger	O
23,25	+12V	+12V	O, power
24	-	Reserved	NA
26	-	Reserved	NA
27	-	Reserved	NA
28	-	Reserved	NA
29	EINTx*	External interrupt	I
30	-	Reserved	NA
31	-	Reserved	NA
32	-	Reserved	NA
33-48	D16-D31	Data bus 16-31	I/O

TABLE 2. **Bus Connector B Pinout**

In addition to the standard signal set, most host cards provide additional processor-specific interface signals on bus connector B which can be used on modules designed specifically for those host cards. See the individual host card’s *Instruction Manual* for additional details on the specific implementation used on a particular card.

Also, please note that certain signals vary by host board and OMNIBUS slot number. For example, the IOMODx signals are unique for each slot on a particular host (on the M44 slot 0 receives IOMOD0-3 while slot one receives IOMOD4-7). Check the *Instruction Manual* for a specific pinout for the particular host board in use.

I/O Connector

The I/O connector provides 50 pins of unspecified connectivity to a(n) external connector(s) on the host board which allow(s) external cabling to be connected to circuitry on the OMNIBUS module. These signals are specific to the particular module in use, and the types of external connectors present on the host board are particular to the host board in use. (See the individual host card's *Instruction Manual* for details on the types of connectors used to make connections to the OMNIBUS I/O pins.)

When using Chico or any Compact PCI card (cM44 or cM62) with any Omnibus modules, it is *VERY IMPORTANT* to note that the standard pin numbering for a SCSI-2 connector causes the Omnibus modules pins to be mapped.

Why did we use this pin mapping? The standard physical numbering of the right angle SCSI-2 connector used on Chico is different from that of the vertical Omnibus connectors. To keep signals clean and separated as much as possible on the Chico, the pins had to be reordered to accommodate the connector and keep the standard pin numbering scheme.

OMNIBUS Memory Mapping and Accesses

OMNIBUS uses memory-mapped accesses from the host board to exchange information with OMNIBUS modules. Four decoded active low module selects are provided per module slot, which are further decodable via the twelve least significant address pins on the bus connectors. The module selects go active low through the host's read or write access into a particular decoded memory region. The amount of memory selected by each module will differs from host to host. Refer to the individual host board's *Instruction Manual* for OMNIBUS memory map details.

OMNIBUS Accesses and Hardware RDY Generation

The host processor's I/O bus control register is initialized for hardware wait states. This means that the individual OMNIBUS modules are responsible for generating an active low ready (RDY) pulse to terminate bus accesses to their respective memory mapped areas. This allows each module to individually determine timing for bus accesses to the memory space in which it is installed. The processors ready input function is shared across both modules as well as the on-board wait-state generation logic, which is responsible for providing ready pulses for on-board I/O space peripherals.

The following example code, written in the AMD MachXL design language, shows how to provide a logic-based wait state generator for customer designed modules. The wait state generator is implemented as a state machine clocked from the OMNIBUS clock signal (this code is used in the MACH210 CPLD device on the OMNIBUS DIG module). The design generates two different levels of wait state length: two wait state cycles for the control registers, digital I/O, and ID ROM accesses and five wait state cycles for all UART accesses.

```
PIN ? H1 COMB

PIN ? /IOMOD0 COMB

PIN ? /IOMOD1 COMB

PIN ? /IOMOD2 COMB

PIN ? /IOMOD3 COMB

PIN ? /LRDY0 REG

NODE ?  LRDY0_TRST REG

NODE ?  Q[2..0] REG


I/O Module bus wait state generator.

CASE (LRDY0_STATE)

BEGIN


 dwell:

  BEGIN

   IF(DUMMY_INPUT) THEN    ;fake IF to fix MACHXL compile bug

    BEGIN

     LRDY0_STATE = 7

     LRDY0 = VCC

     LRDY0_TRST = VCC

    END
```

```
ELSE
 BEGIN
  IF(/IOMOD0 * /IOMOD1 * /IOMOD2 * /IOMOD3) THEN  ;if no cycle, remain in dwell state
   BEGIN
    LRDY0_STATE = dwell
    LRDY0 = GND
    LRDY0_TRST = GND
   END
  ELSE
   BEGIN
    IF(IOMOD0) THEN     ;if UART access, execute five wait cycle
     BEGIN
      LRDY0_STATE = five_wait_start
      LRDY0 = GND
      LRDY0_TRST = VCC
     END
    ELSE
     BEGIN
      IF(IOMOD1 + IOMOD2 + IOMOD3) THEN   ;if any other access, one wait state
       BEGIN
        LRDY0_STATE = one_wait_start
        LRDY0 = VCC
        LRDY0_TRST = VCC
       END
```

```
            END
          END
        END
      END


    one_wait_start:
      BEGIN
     LRDY0_STATE = dwell   ; jump back to DWELL state
     LRDY0 = GND     ; send LRDY0 active low to 'C44
     LRDY0_TRST = GND
      END


    five_wait_start:
      BEGIN
     LRDY0_STATE = five_wait_dwell_one
     LRDY0 = GND
     LRDY0_TRST = VCC
      END


    five_wait_dwell_one:
      BEGIN
     LRDY0_STATE = five_wait_dwell_two
```

```
  LRDY0 = GND

  LRDY0_TRST = VCC

    END



five_wait_dwell_two:

    BEGIN

  LRDY0_STATE = five_wait_dwell_three

  LRDY0 = GND

  LRDY0_TRST = VCC

    END



five_wait_dwell_three:

    BEGIN

  LRDY0_STATE = five_wait_dwell_four

  LRDY0 = GND

  LRDY0_TRST = VCC

    END



five_wait_dwell_four:

    BEGIN

  LRDY0_STATE = five_wait_dwell_five

  LRDY0 = VCC
```

```
 LRDY0_TRST = VCC

  END



five_wait_dwell_five:

  BEGIN

 LRDY0_STATE = dwell   ;jump back to dwell state

 LRDY0 = GND        ;send LRDY0 active to 'C44

 LRDY0_TRST = GND

  END

END



Q[2..0].CLKF = /H1

Q[2..0].RSTF = RESET


LRDY0.CLKF = /H1

LRDY0.RSTF = RESET

LRDY0.TRST = LRDY0_TRST


LRDY0_TRST.CLKF = /H1

LRDY0_TRST.RSTF = RESET
```

FIGURE 2. OMNIBUS Wait State Generator Example

OMNIBUS Timing

The following diagrams will provided timing information for the OMNIBUS interface. This data is derived from device specifications and has not been factory tested.

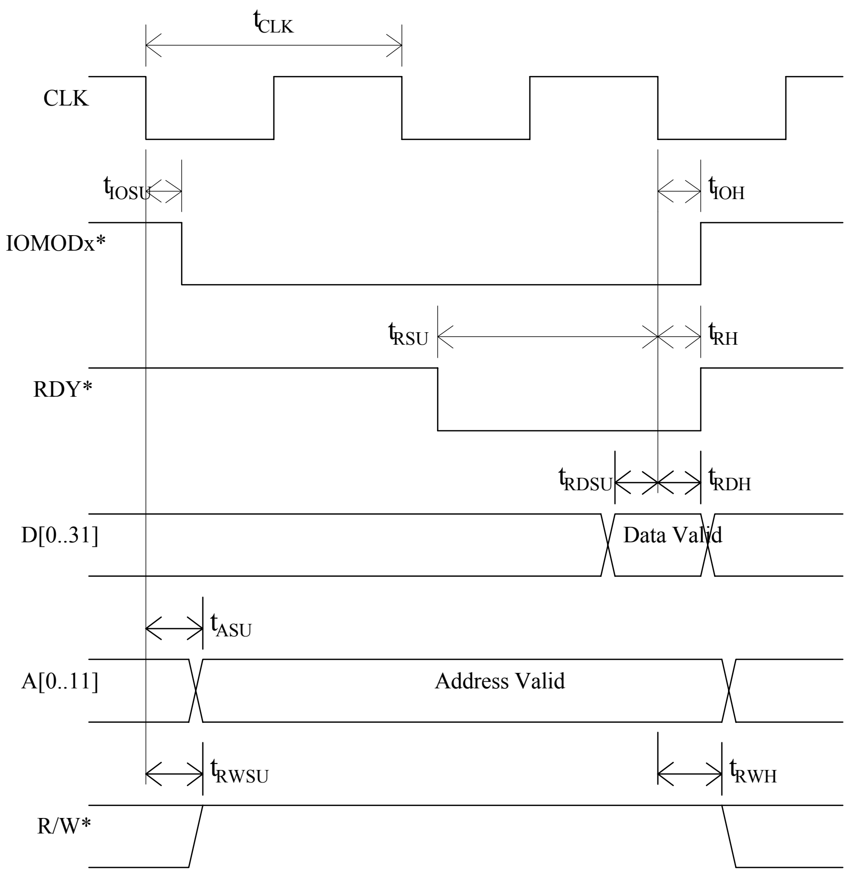


FIGURE 3. OMNIBUS Read Cycle Timing

Parameter	**min. (ns)**	**max. (ns)**
t_{CLK}	33	
t_{IOSU}	0	15
t_{IOH}	0	15
t_{RSU}	20	
t_{RH}	0	
t_{RDSU}	10	

Parameter	min. (ns)	max. (ns)
t_{RDH}	0	
t_{WDSU}		16
t_{WDH}	15	
t_{ASU}		9
t_{RWSU}		9
T_{RWH}		9

TABLE 3. OMNIBUS Read Cycle Timing Parameters

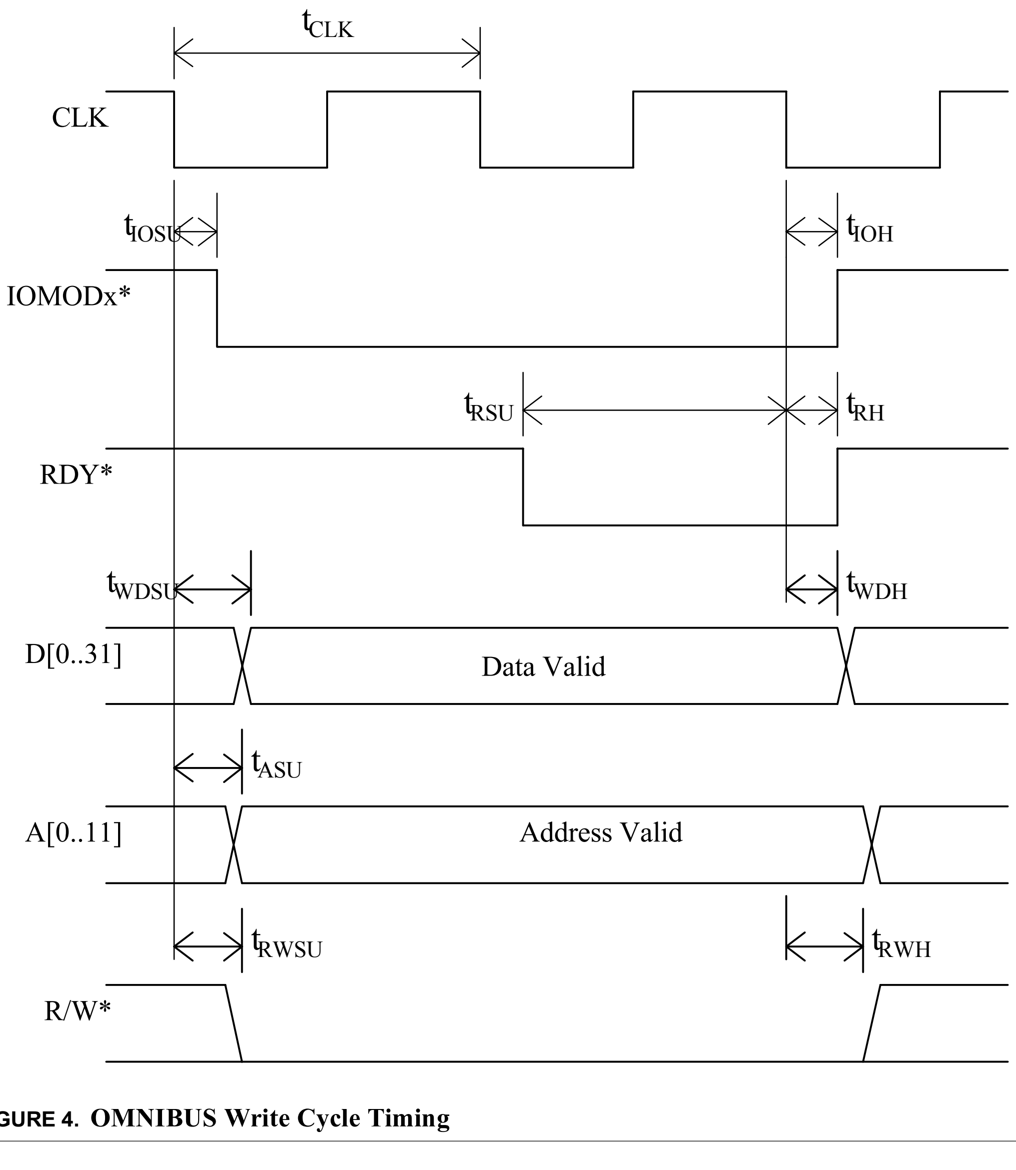


FIGURE 4. OMNIBUS Write Cycle Timing

OMNIBUS Power

The OMNIBUS interface provides five separate power supplies for use by modules along with two separate grounded return connections. The following table lists the power supplies and their power ratings. A separate digital 5V-power supply is provided with a separate digital ground to minimize the digital noise present on the analog power supplies.

Pin Name	Voltage	Current Rating (max.)
DVCC	5V (digital)	(Host system dependent)
+12	12V	(Host system dependent)
-12	-12V	(Host system dependent)
+5V	5V (analog)	500 mA
–5V	-5V	500 mA
+15V	+15V	330 mA
–15V	-15V	330 mA

TABLE 4. **OMNIBUS Power Ratings**

Please note that the AGND and DGND busses are separated on the OMNIBUS host cards. For proper ground referencing, they must be tied together on modules that use the analog power supplies (any supply other than digital 5V, 12V, or –12V). Innovative Integration recommends that a ferrite bead (Panasonic EXC-ELSA35V or equivalent, depending on total current draw) be used on custom modules to connect the two ground busses. This is to prevent high frequency digital noise on the DGND bus from polluting or contaminating the clean AGND return.

OMNIBUS Mechanical Specifications

The following figure gives the mechanical specifications for board size, connector placement and orientation for the I/O bus modules.

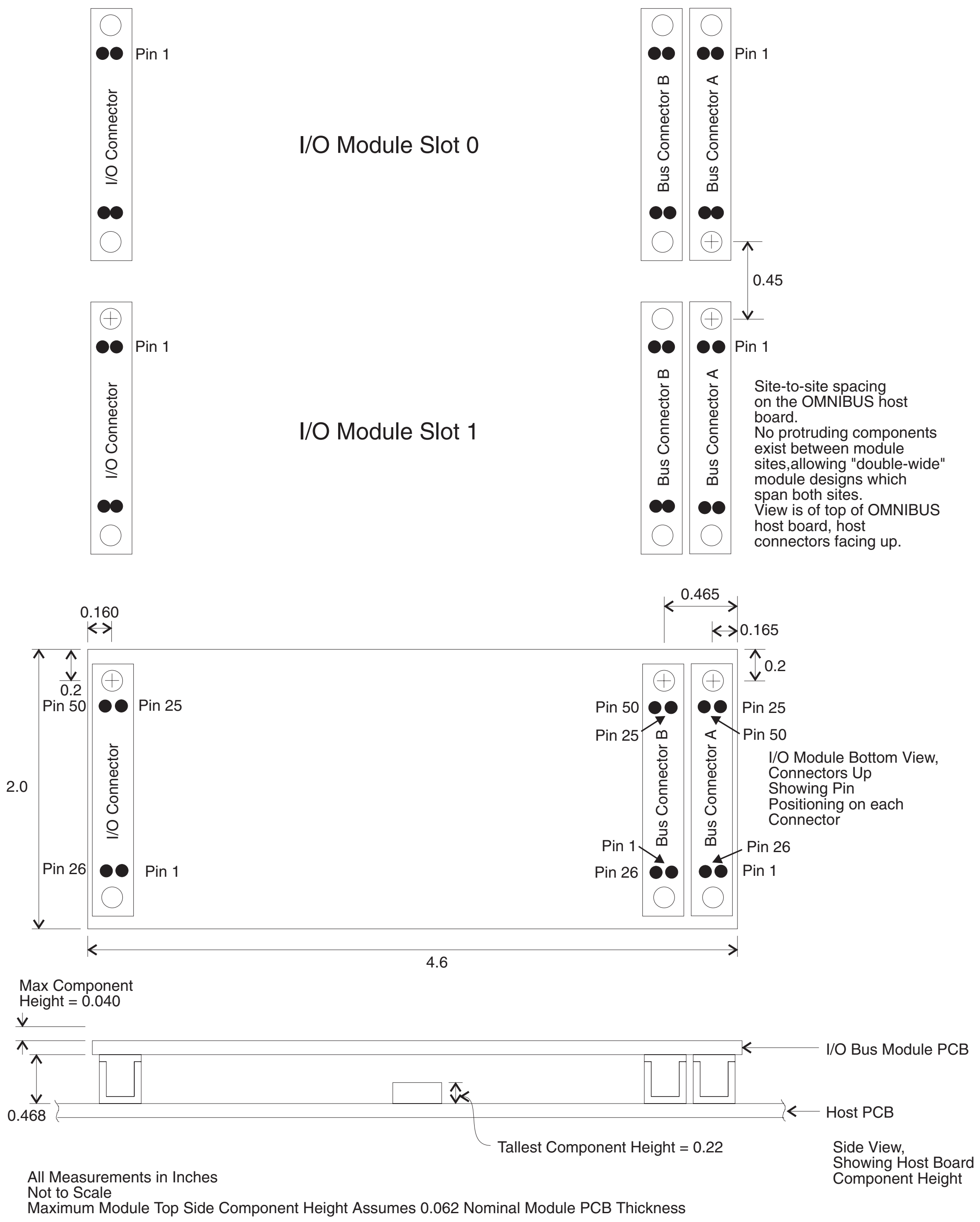

Pin 1
I/O Connector
I/O Module Slot 0
Bus Connector B
Bus Connector A
Pin 1
0.45
Pin 1
I/O Connector
I/O Module Slot 1
Bus Connector B
Bus Connector A
Pin 1
Site-to-site spacing on the OMNIBUS host board.
No protruding components exist between module sites,allowing "double-wide" module designs which span both sites.
View is of top of OMNIBUS host board, host connectors facing up.
0.160
0.465
0.165
0.2
0.2
Pin 50
Pin 25
2.0
I/O Connector
Pin 26
Pin 1
Pin 50
Pin 25
Bus Connector B
Bus Connector A
Pin 25
Pin 50
I/O Module Bottom View, Connectors Up Showing Pin Positioning on each Connector
Pin 1
Pin 26
Pin 26
Pin 1
4.6
Max Component Height = 0.040
I/O Bus Module PCB
0.468
Host PCB
Tallest Component Height = 0.22
Side View, Showing Host Board Component Height
All Measurements in Inches
Not to Scale
Maximum Module Top Side Component Height Assumes 0.062 Nominal Module PCB Thickness

FIGURE 5. **OMNIBUS Mechanical Specification**

OMNIBUS Module Design Guidelines

The following guidelines should be observed when designing a custom module for the OMNIBUS:

1. Keep all bus line loading to two (2) CMOS loads per pin on each module. If more than two device loads are present on an address or data bus lines, buffer the lines with fast (FCT or ABT family) logic.

2. Do not decode the OMNIBUS decoded strobes further using lower order address lines unless address and data latching is included for write accesses to the card. The strobe signals already represent one propagation delay through a set of decode logic on the card (max 7-10 ns). Additional decoding may drive the resultant strobe trailing edge too far past the close of valid data and address information from the host. This can result in either failed or intermittent write operations.

3. Keep the bus signal trace lengths to the absolute minimum. Interface logic should be placed as close as possible to the bus connector end of the module.

4. Treat the bus clock signal with special care. Termination may be required to maintain the signal integrity.

OMNIBUS Compatibility Notes

The following are processor board compatibility issues pertaining to the use of certain OMNIBUS module designs with particular host boards. Please make note of these limitations when designing custom modules, as they may affect portability across various Innovative Intergration’s host boards.

1. C44 Comm. Port vs. ‘C6201 Serial Port – the M44 processor board implements ‘C44 comm. port connections between the processor and each module port to facilitate the design of the comm. port based communications. Since the ‘C6201 processor does not implement communication ports on the ‘C6201, the buffered serial ports have been connected to OMNIBUS sites in a similar fashion to allow for serial port based interfacing to the processor where necessary. The three types of communications systems are incompatible and modules designed to interface to the M44 via a processor comm. port will not function on the ‘C6201. Similarly, modules designed to use the buffered serial port cannot be used on the M44.

Notes on OMNIBUS Memory Mapping and Access Width

‘C6x Host Boards

Please note that the mnemonic addresses given in the following module descriptions assume integer pointer offsets, and must be treated with some care since the ‘C6201 implements byte addressing. OMNIBUS modules should always be addressed using integer (32-bit) pointers: the bus does not support 8-bit or 16-bit accesses by the ‘C6201 processor. Any CPU operation, which causes an IOMODx strobe to go active, will transfer a full 32-bit word to or from the module’s circuitry.

For example, the description of the DIG module notes that the byte (3) direction control register for a module installed in site (0) is mapped to address IOMOD2 + 3. This address should be literally interpreted as 0x156000C, where IOMOD2 is equal to 0x1560000 and the offset adds decimal 12 (three 32-bit words of offset). IOMOD2 + 3 should NOT be interpreted as 0x1560003, since the offset is three (3) 32-bit words, not three (3) bytes.

This addressing is most easily handled in C language by using integer pointers and integer pointer arithmetic, which will always result in the required address alignment. For example, the following code defines a pointer and accesses the byte (3) direction control register with the documented offset:

```
unsigned int *pointer = 0x1560000;

*(pointer + 3) = 0x0;        /* set byte 3 to output mode */
```

The actual accessed memory location is 0x156000C, due to the way pointer math is handled in C.

CHAPTER 2

A4D1 Module

Module Introduction

The A4D1 OMNIBUS module provides the target card processor with four channels of very high speed 10 MHz, 14-bit resolution analog input to digital output conversion (A/D). In addition to a single channel of very high speed 10 MHz, 14-bit resolution digital input to analog output conversion (D/A). This makes it well suited for high-speed data acquisition applications, transient capture, data processing, and control systems. The A4D1 module uses two pairs of Analog Devices AD9240 A/Ds and one AD9774 D/A to provide for excellent dynamic range over a wide input bandwidth.

The A/D's use a novel four stage pipelined architecture as well as a wideband sample-and-hold amplifier making them well suited for direct IF down conversion extended to 45 MHz. The A/D delivers 10 MHz data from a pipeline, which is only four samples deep, resulting in low data latency. In addition, each A/D channel has gain/offset error adjustment for accurate measurements.

A one (1) kword FIFO on each channel separates the A/Ds and D/A from the data bus allowing the DSP time for other tasks while the FIFOs are filling or emptying. The FIFOs allow the DSP to collect and transport the data from the A/D's as single points or as a data set of up to 1K sample size. This reduces the interrupt rate to the host DSP allowing for highly efficient data connection.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

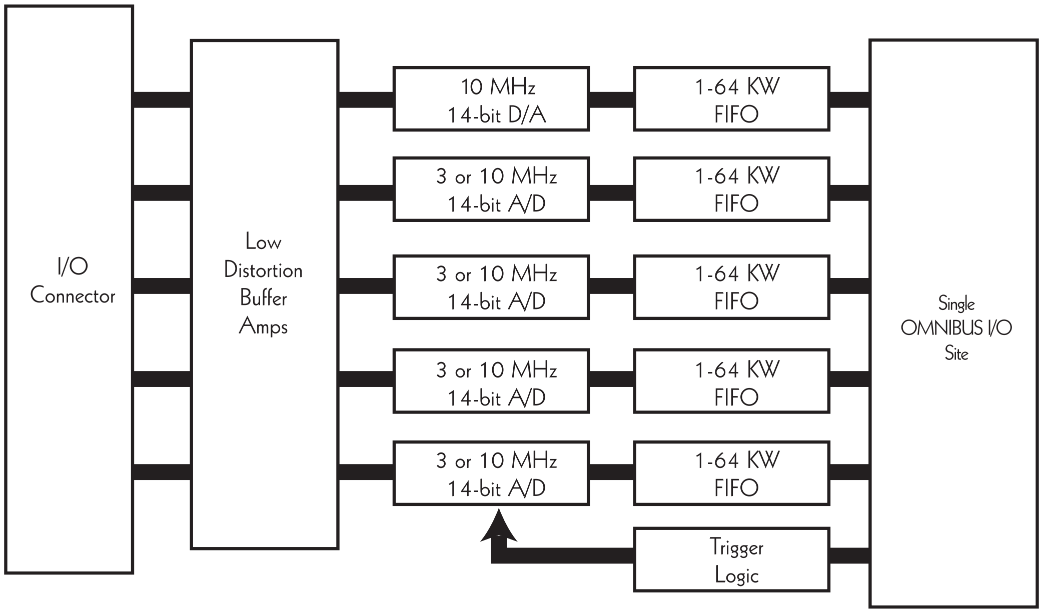


FIGURE 6. **A4D1 Block Diagram**

Bus Type:	Compatible with all Omnibus I.I. Products; 32-bit. Consumes one interrupt to host dsp. Wait-states depend on host platform.	**Filter Characteristics:**	24-pole filter -3 dB set at 5 MHz no overshoot
Power Requirements:	5 V @ 500 mA; +15V @ 20mA; -15 V @ 70mA	**Input Impedance:**	50 ohm
Physicals:	OMNIBUS mezzanine card; 2.0” X 4.6”	**Conversion Trigger Sources:**	DSP, timers or externally triggered
Weight:	0.6 Lbs.	**Interface to DSP:**	Memory-mapped 32-bit result returned for each A/D pair.
A/D Converters: (four A/D chips)	Analog Devices AD9240 Pipelined architecture for low data latency. Each converter channel has independent filtering.	**D/A Converter:**	One analog device AD9774 D/A channel has independent filtering, gain and trim.
Resolution:	14-bit	**Resolution:**	14-bit w/4X interpolation
Update Rate:	10 MHz	**Output Range:**	+/- 5 V custom ranges may be special ordered.
Analog Input Range:	+/- 2 V, custom ranges may be special ordered	**Settling Time:**	35 ns to 0.025%
S/N Ratio:	75 dB	**Dynamic Range:**	96 dB
THD:	80 dB	**Offset Error:**	Trimmable on each channel - factory calibrated to +/- 4 LSB
Dynamic Range:	80 dB	**Gain Error:**	Trimmable on each channel - factory calibrated to +/- 4 LSB
Gain Error:	Trimmable on each channel - factory calibrated	**Diff. Nonlinearity Error:**	+/- 3 LSB - Monotonic
Diff. Linearity Error:	+/- 1 LSB	**D/A Glitch Energy:**	5 pV-sec typical at MSB transition
Offset Error:	Trimmable on each channel - factory calibrated	**Interface to DSP:**	Memory-mapped; 14-bit interface to DSP
Aperature Delay:	1 ns	**Output prop delay:**	55 clocks
Aperature Jitter:	4 ps	**THD:**	-70 dB
Input Type:	Single Ended		

The target Peripheral Library provides support for each of the A4D1 functions, and the following sections give descriptions of the support routines. Refer to the **a4d1adc.h**, **a4d1dac.h** or **aixint.h** files located in the `c:/<target board>/Include/Target/Analog` director and **Omnibus.hlp or Zuma.hlp** Windows helpfile for complete details on each function.

Function Name	Operation
timebase()	Sets sample rate of A/Ds using DDS.
A4D1_initialize()	Resets module to default state.
A4D1_read_adc_pair()	Reads A/D converter sample results from two channels.
A4D1_write_dac()	Writes value to D/A channel FIFO.
A4D1_enable_fifo()	Gates acquisitions from selected A/D pair into FIFO.
A4D1_gate()	Gates acquisitions to all A/D channels.
A4D1_set_gate_polarity()	Configures polarity of gating inputs.
A4D1_reset_fifo()	Resets the A4D1 FIFOs.
A4D1_bleed_fifo()	Reads FIFO contents to a block of memory.
A4D1_fill_fifo()	Writes block of memory into FIFO.
A4D1_trigger_adc()	Configures A/D pair conversion source.
A4D1_trigger_dac()	Configures D/A conversion source.
A4D1_set_dac_bipolar()	Configures D/A output mode.
A4D1_set_dac_output()	Enables/disables D/A output.
A4D1_set_fifo_interrupt_level()	Selects FIFO level interrupt to processor (i.e. half full).
A4D1_read_status()	Read back of FIFO interrupt flags.
A4D1_write_idrom()	Write identification information.
A4D1_read_idrom()	Read identification information.
A4D1_correct_adc_pair()	Corrects adc value to have the proper range.
A4D1_correct_dac()	Corrects dac value to have the proper range.
A4D1_fast_omnibus()	Programs Omnibus for fast FIFO accesses. (For M6x only)
A4D1_normal_omnibus()	Programs Omnibus for normal FIFO accesses. (For M6x only)

TABLE 5. C Language A4D1 Functions

The A4D1 library functions can be divided into several groups:

1. Sample rate control.
2. A/D control.
3. A/D FIFO control.
4. D/A control.
5. D/A FIFO control.
6. Identification readback.

Sample Rate Control

The sample rate to the analog hardware onboard the A4D1 may be precisely controlled via the baseboard's 9850 DDS timer. The **timebase()** function controls the sampling rate used by all the A/D converters and the D/A converter within the application. Alternately either of the baseboard PIT timer channels (PIT0 or PIT1) may be used to control the sampling rate. To achieve synchronization with external events, the A4D1 can be configured to receive conversion clocks from an EXTERNAL source (within the **A4D1_trigger** functions).

A/D Control

The A/Ds on the A4D1 begin sampling data after the conversion clock begins running. The functions **A4D1_enable_fifo()** and **A4D1_gate()** control the gating of data.

A/D FIFO Control

To begin storing sampled data in the FIFOs, both the **A4D1_enable_fifo ()** & **A4D1_gate()** functions must be called. The gate function allows data to begin entering all of the ADC and DAC FIFOs simultaneously on previously triggered pairs. The FIFOs will continue to fill until they are full at which point no more data will be stored until the FIFOs are reset or emptied.

The FIFOs can be set up to interrupt or flag the processor at three points; full, half full, or not empty using the **A4D1_set_fifo_interrupt_level()** function and the state of these FIFO flags can be read at any time with the **A4D1_read_status()** function.

To read the converted data from the FIFOs, use the **A4D1_read_adc_pair()** function, which takes the pair number to read as an argument (pair 0 for channels 0 & 1, pair 1 for channels 2 & 3). The data comes in stacked on the 32-bit bus with the lower 16 bits being the first channel of the pair and the high 16 bits being the second channel of the pair. The function compensates for any analog inversion on the A/D front end and for the offset-binary output format of the A/D devices.

The FIFOs may be cleared using the **A4D1_reset_fifo()** function.

D/A Control

The D/A on the A4D1 begins sampling data after the conversion clock begins running. The functions **A4D1_enable_fifo()** and **A4D1_gate()** control the gating of data.

D/A FIFO Control

To begin reading data previously written into the D/A FIFO, both the **A4D1_enable_fifo()** & **A4D1_gate()** functions must be called. The gate function allows data to begin entering all of the ADC and DAC FIFOs simultaneously on previously triggered pairs. The FIFO will continue to deplete until it is empty at which point the same data point data will be read by the D/A until the FIFO is refilled.

The FIFO can be set up to interrupt or flag the processor at three points; full, half full or not empty using the **A4D1_set_fifo_interrupt_level()** function and the state of these FIFO flags can be read at any time with the **A4D1_read_status()** function.

To store data into the FIFO, use the **A4D1_write_dac()** function, which takes the value to write as an argument. The data should be standard two's complement, with the lower 14 bits being significant. The function compensates for any analog inversion on the D/A front end and for the offset-binary output format of the D/A device.

The FIFO may be cleared using the **A4D1_reset_fifo()** function.

Identification Readback

The **A4D1_read_idrom()** function is used to read the identification ROM on the A4D1 to check its identity and revision level. The function fills out an A4D1_ID structure with the information stored in the ID ROM on the module. The A4D1_ID structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string "A4D1").
2. Module revision level (single character revision level).
3. Checksum.

This data is preprogrammed at the factory and should not be altered by the user.

Example DSP Target Programs for the A4D1 Module

The following sections describe the example programs available in the Developer's Package for use with a target DSP card with an A4D1 analog interface module installed.

TEST: `TEST` utilizes the target DSP's A4D1 module analog input circuitry to sample an external signal and calculate statistics on the resulting digital data. This program is an example of how to handle interrupt driven A/D sampling at high rates, and can serve as a test program for determining the statistical noise performance of a single A/D channel.

`TEST` uses the target DSP trigger matrix electronics to set up an external timer to trigger conversions on a selected A/D converter channel. As conversions are triggered, the A/D's conversion complete interrupt causes an external interrupt on the processor. This causes the interrupt handler to run, which retrieves the newly converted data and stores it to a buffer in processor memory. Once the buffer is full, the interrupt is deactivated and the program code proceeds to calculate the statistics variables on the gathered data.

For Codewright Users: For Codewright Users, the linker command (`.CMD`) files and Codewright project (`.PJT`) files necessary to rebuild the `TEST` program are included with the Developer's Package.

For Code Composer Studio Users: For Code Composer Studio User, the make file (`.MAK`) file necessary to rebuild the TEST program are included with the Developer's Package.

`TEST` may be used as a basis for a custom data acquisition program, handling one or more A/D channels and either buffering the required data or passing the data to a host program via the card's bus mastering capability (see the appendices for more information about host applications).

Memory Mapping

The A4D1 module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** or the **a4d1.h** file included with the Zuma Tools DSP software library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	Read/ Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address	Bit Field Value
A/D Pair 0 Data Read	R	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0	
A/D Pair 0 Enable FIFO Fill	W	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0	1 Enables, 0 Disables*
A/D Pair 1 Data Read	R	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1	
A/D Pair 1 Enable FIFO Fill	W	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1	1 Enables, 0 Disables*
Gate Enable	W	IOMOD0 + 0x2	IOMOD4 + 0x2	IOMOD8 + 0x2	1 Enables all FIFOs (A/D & D/A) simultaneously, 0 – Disables*
FIFO Interrupt Flag Selection	W	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3	Bit 0 – Not Empty Pair 0 Bit 1 – Half Full Pair 0* Bit 2 – Full Pair 0 Bit 3 – Not Empty Pair 1 Bit 4 – Half Full Pair 1* Bit 5 – Full Pair 1 Bit 6 – Empty DAC Bit 7 – Less than Half Full DAC* Bit 8 – Not Full DAC

Function	Read/ Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address	Bit Field Value
Interrupt Flag Readback	R	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3	Bit 0 – Not Empty Pair 0 Bit 1 – Half Full Pair 0 Bit 2 – Full Pair 0 Bit 3 – Not Empty Pair 1 Bit 4 – Half Full Pair 1 Bit 5 – Full Pair 1 Bit 6 – Empty DAC Bit 7 – Less than Half Full DAC Bit 8 – Not Full DAC
FIFO Reset	W	IOMOD0 + 0x4	IOMOD4 + 0x4	IOMOD8 + 0x4	Bit 0 – write 1 to reset A/D FIFO's Bit 1 – write 1 to reset D/A FIFO's
Clock Selection Matrix	W	IOMOD0 + 0x8	IOMOD4 + 0x8	IOMOD8 + 0x8	Bit 0 – A/D clock = DDS* Bit 1 – A/D clock = TMR0 Bit 2 – A/D clock = TMR1 Bit 3 – A/D clock = EXT A/D CLK Bit 4 – D/A clock = DDS* Bit 5 – D/A clock = TMR0 Bit 6 – D/A clock = TMR1 Bit 7 – D/A clock = EXT D/A CLK
D/A Output Selection	W	IOMOD0 + 0xA	IOMOD4 + 0xA	IOMOD8 + 0xA	Bit 0 – 0 – Bipolar*, 1 – Unipolar Bit 1 – 1 – Output Connected, 0 – Output Shorted to GND*
External Gate Polarity	W	IOMOD0+ 0xC	IOMOD4+ 0xC	IOMOD8+ 0xC	Bit 0 – Ext Start Polarity 0 – Rising edge sensitive* 1 – Falling edge sensitive Bit 1 – Ext Stop Polarity 0 – Falling edge sensitive* 1 – Rising edge sensitive
Empty D/A FIFO	W	IOMOD0 + 0xE	IOMOD4 + 0xE	IOMOD8 + 0xE	1 –Empty D/A FIFO, 0 –Disable D/A FIFO Output
Fill D/A FIFO	W	IOMOD0 + 0xF	IOMOD4 + 0xF	IOMOD8 + 0xF	Write Fills the D/A FIFO
IDROM SDA	R/W	IOMOD3 + 0x0	IOMOD7 + 0x0	IOMOD11 + 0x0	
IDROM SCK	W	IOMOD3 + 0x1	IOMOD7 + 0x1	IOMOD11 + 0x1	
Gate Enable	W	IOMOD0 + 0x2	IOMOD4 + 0x2	IOMOD8 + 0x2	1 Enables all FIFOs (A/D & D/A) simultaneously, 0 – Disables*

TABLE 6. A4D1 Memory Map (* Default at RESET)

Interrupt Usage

The A4D1 uses a single external interrupt to the baseboard processor to signal the appropriate FIFO flag. To determine which flag caused the interrupt, the flag register must be read back. This interrupt may be used to trigger CPU interrupts or to begin a DMA to transfer data from the FIFO to local or global memory.

The type of FIFO interrupt given to the processor is selectable as shown in above memory map table. Typically, users will want to use the half full interrupt (default at power up) so ample time is available to begin moving data without gaps between the receipt of the flag and the start of data movement.

As an example:

Writing a 0x2 to the interrupt flag selection address will give the user an interrupt when the FIFO is half full. Reading back the interrupt flags from the same address tells the status of all the FIFO's so the user can determine which FIFO's are interrupting the processor.

Writing a 0x80 to the interrupt flag selection address will give the user an interrupt when the DAC FIFO is less than half full. This condition can also be read back to distinguish between the A/D and D/A interrupts

The A4D1 module can be installed in Module site 0 or 1. This allows the card to function properly with the interrupt jumpers in their default position on the target card since the A4D1 module uses external interrupt 0 for Module site 0 and external interrupt 2 for Module site 1.

Pin Connector I/O

The A4D1 output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* for additional details on how to connect to the A4D1's I/O pins.

A4D1 Omnibus M173280-3 Connector Pin Number	**ChicoPlus & Hombre Baseboard 100 Pin MDR Connector**		**cM44 & cM6x Baseboard SCSI-2 50-Pin Connector**	**M44, M6x & SBC6x Baseboard IDC50 Connector**	**M44 & M6x Baseboard DB15 Connector**	**Function**
	Module Site 0	**Module Site 1**				
1..35	34..50, 83..100	9..25, 58..75	1..25, 41..50	1..35		Reserved
36	33	8	40	36	1	AGND
37	82	57	39	37	9	Channel 0 Input
38	32	7	38	38	2	AGND
39	81	56	37	39	10	Channel 1 Input
40	31	6	36	40	3	AGND
41	80	55	35	41	11	Channel 2 Input
42	30	5	34	42	4	AGND
43	79	54	33	43	12	Channel 3 Input
44	29	4	32	44	5	AGND
45	78	53	31	45	13	DAC Output
46	28	3	30	46	6	DGND
47	77	52	29	47	14	External End
48	27	2	28	48	7	External A/D Clock
49	76	51	27	49	15	External Start
50	26	1	26	50	8	External D/A Clock

TABLE 7. A4D1 I/O Connector Pinout

Functions

Analog Input

The Input to the A4D1 has a 50-ohm impedance to ground allowing the user to use shielded cable to preserve signal integrity and maintain minimal cross talk between channels. The input buffering is done with wide band low distortion amplifiers maintaining the low signal to noise and distortion of the A/Ds.

The analog to digital converters (AD9240) are 14 bit devices with a maximum output rate of 10 MHz capable of down converting signals as high as 45 MHz The A/Ds are stacked in pairs on the data bus to allow two channels to be read in one cycle and have an input range of 4 Volts Pk-Pk. For more in depth information on the AD9240 refer to the data sheet, which can be found on Analog Devices website (www.analog.com).

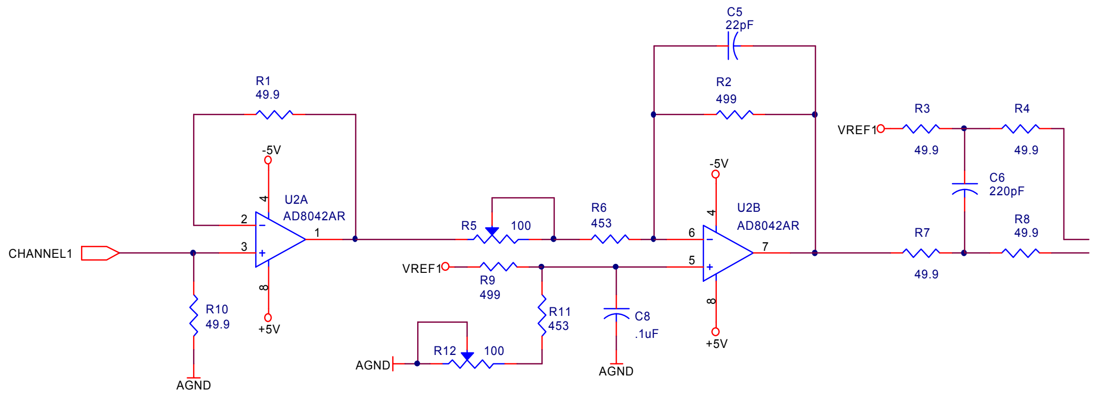


FIGURE 7. **A4D1 Input Schematic**

Analog Input Conversion Triggering

The A/D converters are triggered by the output of the DDS, TMR0, TMR1 or an external clock. As soon as the clock has been setup in the software (see Target board Development Package Manual) to output a clock at a specified rate, the A/Ds begin converting data. With this particular A/D there is a three clock pipelined delay before the data is output.

The data is not stored until both the FIFO and the Gate has been enabled for a pair of channels. The Gate allows data gathering on all four channels to begin simultaneously, while the FIFO enable lines can allow a specific pair to be enabled at a time provided the Gate signal has already been enabled.

The gate function may be executed from either software or external TTL signals and both may be used in conjunction with one another. For example an external signal on the EXT START pin can begin the storing of data and when a predefined number of points have been read in, the data can be inhibited via the software commands. The inverse is also true, data can begin storage from a software command and stopped from an external signal on the EXT STOP pin. All the signals are edge triggered events and are not level sensitive. The polarity of the external signals is also programmable allowing the user to tie the two signals together as one control signal.

Analog Input Trimming

The ADC offsets and gain are set at the factory, but are user trimmable with the following procedure. Errors should be trimmed at the expected normal operating temperature. The trim pot locations are given in the table below.

1. Short the inputs for each channel to AGND on connector P1. You may want to do this out at your sensor to include any error sources present in the application circuitry. Trim offset pots such that a zero code is observed when continuously sampling the converters.

2. Connect one input to a voltage reference and check gain for proper codes. Full scale inputs, at a gain of 1, are +/-2V and should read raw ADC codes of 8192 and -8192 respectively.

Channel	Gain	Offset
0	R5	R12
1	R19	R26
2	R33	R40
3	R47	R54

TABLE 8. **A4D1 Gain & Offset Adjustment**

Analog Output

One channel of 14-bit high speed digital-to-analog conversion is provided on the A4D1. The D/A is useful for analog signal outputs for both control and signal generation.

Digital data is written to the D/A for output via a memory mapped location. The data bus is connected to the D/A occupying the lower 14 bits of the bus at address IOMOD0 + 0xF or IOMOD4 + 0xF depending on which Module site the card is installed in (see memory map table).

The data written to the D/A devices is in straight binary format, rather than two’s complement. The non-inverting output stage of each D/A channel yields a [-5V .. +5V] output voltage for an input code range of [0..16384]. When in bipolar mode (see memory map table), yields a [0…+5V] output voltage for the same input code range when in unipolar mode.

There is also an analog switch that shunts the output to ground until it is enabled via software. This allows the user a predefined power up state of the output, which can be helpful in control applications.

Analog Output Conversion Triggering

The DAC is continuously updated via the 80MHz clock on the A4D1 module and new data is clocked out of the DAC FIFO using the DDS, TMR0, TMR1 or the external D/A clock. This is done to allow the 4x interpolation filter to work on the DAC while allowing an update rate from the processor up to 10 MHz. Simply stated, the output of the DAC is continuously updated at 20 MHz and the input is updated at the desired playback rate. Refer to the memory map table for the proper clock selection matrix.

The DAC FIFO can be programmed to give the processor an interrupt at either NOT FULL, less than HALF FULL, or EMPTY. The default is less than HALF FULL, allowing the FIFO to be used as a 512 point data buffer, which can be continuously filled while data is being output. The FIFO begins outputting data synchronously with the chosen DAC clock after both the output and the gate has been enabled. The gate allows for simultaneously starting A/D conversion as well as the D/A conversion.

Analog Output Buffer

A schematic of the analog output buffer is shown on the figure below. The DAC provides two complimentary current outputs whose full scale current is 20 mA. The differential voltage developed across the DAC outputs is converted to a single ended voltage via the differential op-amp configuration.

The output from the op-amp, OUT1 is connected via an analog switch to the output connector. In Bipolar mode OUT2 and IN2 are not connected to anything and the output to the connector is +/-5V. In Unipolar mode, OUT1 is connected to OUT2 and IN1 is connected to IN2 resulting in an output to the connector of 0-+5V. C89 forms a single pole filter to reduce distortion by preventing the DAC output from overloading the op-amp's input.

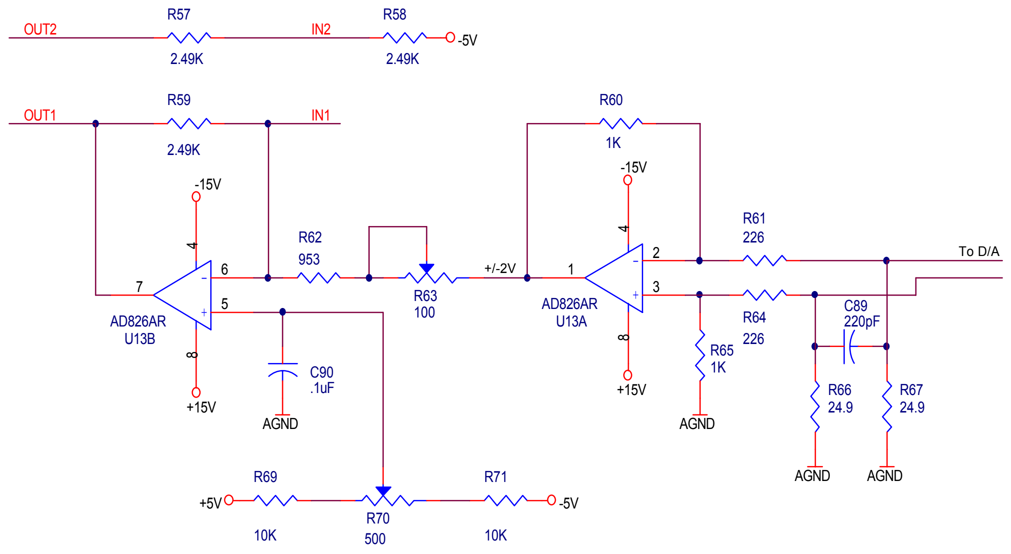


FIGURE 8. A4D1 Analog Output Buffer

CHAPTER 3

A4D4/TERM Modules

Module Introduction

The A4D4 OMNIBUS module provides the target card processor with four channels of high speed 200 kHz, 16-bit resolution analog input to digital output conversion (A/D) per module slot. In addition, four channels of high speed 200 kHz, 16-bit resolution digital input to analog output conversion (D/A). The A4D4 has analog I/O that is tightly coupled with the DSP, making it well suited for controls systems, process monitoring, and data acquisition applications. The A4D4 module uses two pairs of Analog Devices AD976AA A/Ds with each channel having independent input six-pole anti-alias filters and programmable gain amplifiers provide for flexible input. While two pairs of Analog Devices AD7846 D/As with output amplifiers and independent channel filtering, gain, and trim, provide for high speed data output signals.

The four analog inputs on the A4D4 module are successive approximation type A/D converters, which allows for low data latency that is critical in control applications and multiplexed channel configurations. In addition, each A/D channel is calibrated for offset and gain errors allowing accurate measurements for a variety of applications. The converters may be triggered via hardware timer or software access and are capable of interrupting the target processor in interrupt driven applications.

The TERM card may be purchased to expand each of the A/D input with a multiplexor for a total of 32 single-ended channels or 16 differential channels of input to each A4D4 module. The TERM card is controlled by the OMNIBUS host DSP card so that the channel indexing is under software control.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

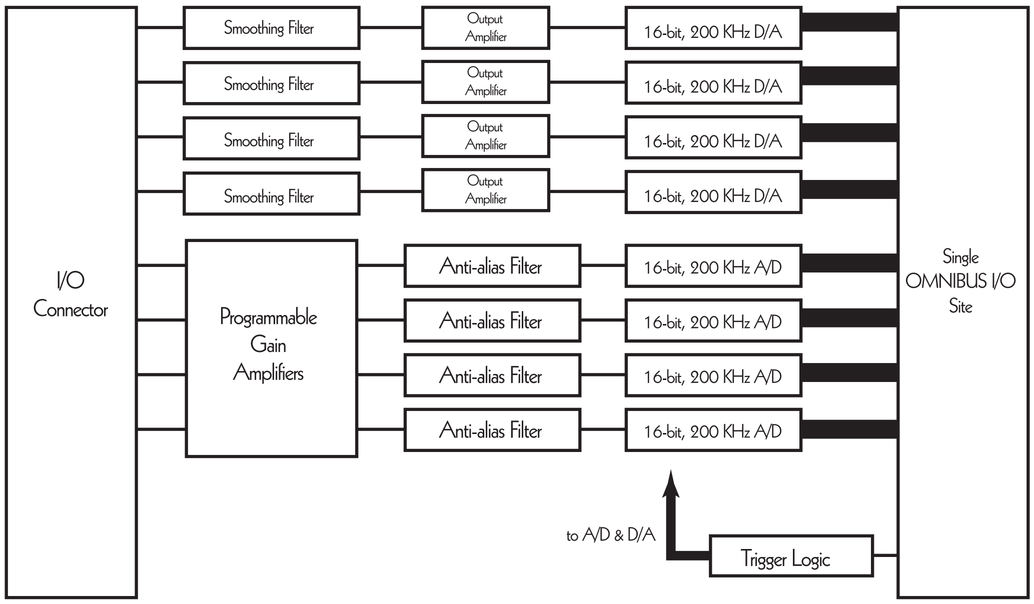


FIGURE 9. **A4D4 Block Diagram**

Bus Type:	Compatible with all I.I. products; 32-bit. Consumes one interrupt to host DSP. Wait-state depends on host platform.	**Programmable Gain:**	1,2,4,8 standard; 1,2,5,10 available special order
Power Requirements:	5 V @ 130mA analog; 5 V @ 160 mA, +15V @ 30 mA, -15V @ 30mA	**Input Type:**	Differential
Physicals:	OMNIBUS mezzanine card 2.0” X 4.6”	**Input Impedance:**	1M | | 5 pF; Each input leg is tied to ground with 1 M resistor.
A/D Converters: (four A/D chips)	Analog Devices AD976AA Successive approximation architecture for low data latency. Each converter channel has independent filtering and programmable gain.	**Filter Characteristics:**	6-pole elliptic filter -3 dB set at 100 kHz no overshoot
Resolution:	16-bit	**Conversion Trigger Sources:**	DSP, timers or externally triggered.
Update Rate:	200 kHz	**Interface to DSP:**	Memory-mapped 32-bit result returned for each A/D pair
Settling Time:	5 us (no filtering) @ 10V step to 0.0008%	**D/A Converter:**	Four Analog Devices AD7846. Each D/A channel has independent filtering, gain and trims
Analog Input Range:	+/- 10V, +/- 5V, +/- 2.5V, +/- 1.25V, software programable.	**Resolution:**	16-bit
S/N Ratio:	85 - 90 dB	**Output Range:**	+/- 10 V Custom ranges may be special ordered
THD:	-70 dB (improved if filter defeated)	**Settling Time:**	7 us (no filtering) to 0.003%
Dynamic Range:	90 dB	**Dynamic Range:**	96 dB
Gain Error:	Trimmable on each channel - factory calibrated	**Offset Error:**	Trimmable on each channel - factory calibrated to +/- 4 LSB
Diff. Linearity Error:	+ 3 / - 2 LSB	**Gain Error:**	Trimmable on each channel - factory calibrated to +/- 4 LSB
Offset Error:	Trimmable on each channel - factory calibrated	**Diff. Nonlinearity Error:**	+/- 1 LSB - Monotonic
Aperature Delay:	40 ns	**D/A Glitch Energy:**	400 nV-sec typical at MSB transition
Aperature Jitter:	Meets AC specs	**Interface to DSP:**	Memory-mapped; 16-bit interface to DSP

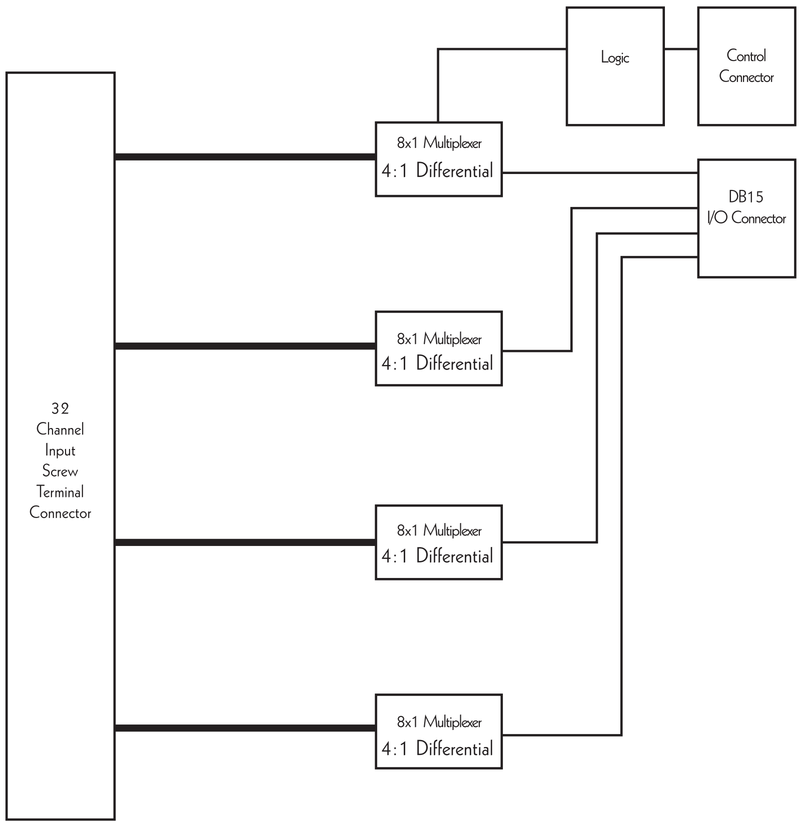


FIGURE 10. TERM Block Diagram

Bus Type:	Compatible with I.I. products TERM interface (Chico, M44, cM44, M6x, cM6x, SBC6x)	**Connectors:**	80 screw terminals for analog I/O and triggers. 14-pin header for digital control inputs. DB9 female for analog outputs.
Power Requirements:	5 V @ 100 mA form M44 mux connector. On-board DC/DC for analog circuitry.	**Outputs:**	Four single-ended outputs to A4D4 module. Four DAC outputs from A4D4 module.
Physicals:	External card connects to DSP with two cables 8.165” X 3.940”. Mounts to DIN rail.	**Analog Inputs:**	32 S.E./16 Diff. inputs; 4 S.E./4 Diff. outputs; Multiplexed 8:1 single-ended or 4:1 differential
Offset Error:	Trimmable with pot on each channel.	**Trigger Inputs:**	Two trigger inputs on terminal block.
Gain Error:	Trimmable with pot on each channel.		

The target Peripheral Library provides support for each of the A4D4 and TERM functions, and the following sections give descriptions of the support routines. Refer to the **adc.h**, **gain.h**, **dac.h**, and **term.h** files located in the `c:/<target board>/Include/Target/Analog` director and the **Omnibus.hlp or Zuma.hlp** Windows help file for complete information on each function.

Function Name	**Operation**
timebase()	Sets sample rate of A/Ds using DDS.
A4D4_read_adc()	Read A/D channel sample results.
A4D4_read_adc_pair()	Read pair of A/D channels as 32-bit value
A4D4_convert_adc_pair()	Trigger A/D conversion on device pair.
A4D4_trigger_adc_pair()	Set A/D hardware trigger source for specified device pair.
A4D4_correct_adc()	Corrects adc value to have the proper range.
A4D4_correct_adc_pair()	Corrects adc pair value to have the proper range.
A4D4_read_dac()	Read the current level of the dac.
A4D4_read_dac_pair()	Read the current level of the dac pair.
A4D4_write_dac()	Write D/A channel data.
A4D4_write_dac_pair()	Write D/A channel data to specified pair.
A4D4_convert_dac_pair()	Trigger D/A converter pair
A4D4_trigger_dac_pair()	Set D/A pair hardware trigger source.
A4D4_correct_dac()	Corrects dac value to have the proper range.
A4D4_correct_dac_pair()	Corrects dac pair value to have the proper range.
A4D4_read_idrom()	Read identification information.
A4D4_write_idrom()	Write identification information.
A4D4_write_gain()	Sets the gain value on the indicated A/D channel.
A4D4_read_gain()	Reads the gain value on the indicated A/D channel.
A4D4_int_select()	Sets the interrupt pin driven by the A4D4.
TERM_set_mux()	Set channel multiplexer.
TERM_set_all_muxes()	Set all multiplexers simultaneously.
TERM_get_mux()	Get the mux input channel on TERM
TERM_sample()	Sample channel.
TERM_reset()	Reset the TERM module.

TABLE 9. C Language A4D4/TERM Functions

The A4D4/TERM library functions can be divided into several groups:

1. Sample rate control.
2. A/D control.
3. D/A control.
4. Interrupt selection.
5. Identification readback.

Sample Rate Control

The sample rate to the analog hardware onboard the A4D4 may be precisely controlled via the baseboard's 9850 DDS timer. The **timebase()** function controls the sampling rate used by all the A/D converters and the D/A converter within the application. Alternately either of the baseboard PIT timer channels (PIT0 or PIT1) may be used to control the sampling rate. To achieve synchronization with external events, the A4D4 can be configured to receive conversion clocks from an EXTERNAL source. The adc and dac's can be independently set by **A4D4_trigger_adc_pair** and **A4D4_trigger_dac_pair** functions respectively

A/D Control

The Peripheral Library includes functions for transferring data from the A/Ds, triggering A/D conversions in software, setting up a hardware timer-based trigger source, and selecting the gain and multiplexer settings on each channel.

A4D4_read_adc() is used to retrieve sample results from one of the A/D devices. The current output buffer contents are returned for the selected channel. If a conversion is in progress, (i.e. a conversion request has been issued to the A/D, either by a software access or by a hardware timer) then the value returned is the last conversion result. The A4D4 architecture stacks two A/D devices on the 32-bit data bus. The **A4D4_read_adc_pair()** function retrieves the previously converted results from the specified device pair. This is more efficient in multi-channel acquisition algorithms.

The A4D4 supports conversion triggers to each pair of A/Ds initiated by either software accesses to the A/D conversion trigger address, or by hardware timebases. There are two A/D conversion trigger signals on each A4D4 module, with each signal tied to a pair of A/D convertors. Thus conversions are always initiated on a pair of A/Ds, and it is not possible to trigger a conversion on only a single A/D device. Software accesses will cause a conversion to start immediately on the selected pair of A/D devices, while hardware triggered conversions are started by a regular timebase "tick" generated at a programmed rate from one of the target timebases. The **A4D4_convert_adc()** routine causes a software timebase conversion, while the **A4D4_trigger_adc()** function programs the A4D4 module's trigger selection matrix to select an target timebase output to trigger the desired A/D pair. Once selected, the hardware timebase must be programmed for the appropriate trigger rate by the **timebase()** function.

Gain and multiplexer control is available with the **A4D4_set_gain()** and **TERM_set_mux()** functions, respectively. These functions select the gain and multiplexer settings for the specified channel. To maximize channel settling time after gain and multiplexer switching, software should set the gain and input channel immediately after a conversion is triggered. This takes advantage of the A/D converter's sample and hold circuitry, which allows the A/D to take an analog sample immediately after a conversion request, which is used to derive the digital conversion value. Once the sample and hold has triggered (within ~40 ns after the conversion request), the input signal may be changed without affecting the A/D's conversion result.

If the TERM module has been purchased, additional sample and hold capability is available which allows up to 8 input channels to be connected to each A/D on the A4D4 for a total of up to 32 channels.

The **TERM_set_mux()** function is provided to set the mux channel selection for each A/D device, and the **TERM_sample()** function triggers an analog sampling operation in the sample/hold circuitry of the TERM. Since the TERM module acts as a combination multiplexer and sample and hold circuit, it is necessary to set the desired input channel (using **TERM_set_mux()**) and trigger a hold operation (using **TERM_sample()**) for the signal selected by the mux before initiating the A/D conversion which will capture the signal in digital form. If software-based A/D conversions are being performed, the correct sequence is as follows:

1. call **TERM_sample()** to trigger the analog sample in the TERM's sample and hold circuit.
2. call **TERM_set_mux()** to set the input channel.
3. delay a few microseconds to allow for multiplexer time to settle to the new signal level.
4. call **A4D4_convert_adc()** to begin an A/D conversion on the new signal.

If hardware based triggers are used, an appropriate calling sequence (usually used in an interrupt routine based on the trigger timebase) is the following:

1. call **TERM_sample()** to trigger the analog sample in the TERM's sample and hold circuit.
2. call **TERM_set_mux()** to set the input channel.

Hardware-triggered sampling has the advantages that it automatically triggers the A/D conversion (no software access is required to the A/D) and that it does not require software to wait for the multiplexer to settle to each new input voltage before initiating a sample (as does software-triggered conversion).

The TERM also supports a single access all devices multiplexer setup mode, which causes all multiplexers on the module to be simultaneously set to the same input channel number. The **TERM_set_all_muxes()** function causes all multiplexer devices to change to the same input number, which is convenient for applications which do continuous sweeping conversions through the same number of inputs on each multiplexer.

D/A Control

The Peripheral Library includes functions for transferring data to the D/As, triggering D/A conversions in software, and setting up a hardware timer-based trigger source.

A4D4_write_dac() is used to send conversion data to one of the D/A devices. The data is latched in the D/A's input buffer, but is not reflected on the analog output of the D/A until a conversion strobe is sent to the device.

As with the A/D converters, the A4D4 supports conversion triggers to each pair of D/As initiated by either software accesses to the D/A conversion trigger address, or by hardware timebases. There are two D/A conversion trigger signals on each A4D4 module, with each signal tied to a pair of D/A convertors. Thus conversions are always initiated on a pair of D/As, and it is not possible to trigger a conversion on only a single D/A device. Software accesses will cause a conversion to start immediately on the selected pair of D/A devices, while hardware triggered conversions are started by a regular timebase “tick” generated at a programmed rate from one of the target timebases. The **A4D4_convert_dac()** routine causes a software timebase conversion, while the **A4D4_trigger_dac()** function programs the A4D4 module’s trigger selection matrix to select an target timebase output to trigger the desired D/A pair. Once selected, the hardware timebase must be programmed for the appropriate trigger rate by the **timebase()** function.

If software-based D/A conversions are being performed, the correct sequence is as follows:

1. call **A4D4_write_dac()** to write the new D/A values to the required channel.

2. call **A4D4_convert_dac()** to begin a D/A conversion and update the output voltage.

If hardware based triggers are used, software only needs to load the D/A registers with the new value prior to a conversion trigger (usually done in an interrupt routine based on the trigger timebase). The conversion will be triggered by the hardware timebase.

Identification Readback

The **A4D4_read_idrom()** function is used to read the identification ROM on the A4D4 to check its identity and revision level. The function fills out an **A4D4_ID** structure with the information stored in the ID ROM on the module. The **A4D4_ID** structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string “A4D4”)

2. Module revision level

3. Checksum

This data is preprogrammed at the factory and should not be altered by the user.

Example Programs for the A4D4/TERM Modules

The following sections describe the example programs available in the Developer’s Package for use with a target DSP card with an A4D4 analog interface module installed.

SNAP. `SNAP` utilizes the target DSP’s A4D4 module analog input circuitry to sample an external signal and calculate statistics on the resulting digital data. This program is an example of how to handle interrupt driven A/D sampling at high rates, and can serve as a test program for determining the statistical noise performance of a single A/D channel.

`SNAP` uses the target DSP trigger matrix electronics to set up an external timer to trigger conversions on a selected A/D converter channel. As conversions are triggered, the A/D's conversion complete interrupt causes an external interrupt on the target processor. This causes the interrupt handler to run, which retrieves the newly converted data and stores it to a buffer in processor memory. Once the buffer is full, the interrupt is deactivated and the program code proceeds to calculate the statistics variables on the gathered data.

For Codewright Users: For Codewright Users, the linker command (`.CMD`) files and Codewright project (`.PJT`) files necessary to rebuild the `SNAP` program are included with the Developer's Package.

For Code Composer Studio Users: For Code Composer Studio User, the make file (`.MAK`) file necessary to rebuild the `SNAP` program is included with the Developer's Package.

`SNAP` may be used as a basis for a custom data acquisition program, handling one or more A/D channels and either buffering the required data or passing the data to a host program via the card's bus mastering capability (see the section below for more information about host applications).

WAVE. `WAVE` is very similar in organization to the `SNAP` program, discussed above, and serves as an example of using the D/A converter outputs to generate signals based on digital signal data buffered in the DSP's memory.

The program first uses the C compiler's trigonometric functions to generate a scaled sine wave table in memory, in a format suitable for use by the D/A devices. The program sets up an external timer as a timebase for D/A conversions, and initializes the target DSP's trigger matrix to direct the timebase output to the D/A devices. This causes the D/A's to continuously convert the values present in their data latches. Simultaneously, the conversion trigger also causes an analog interrupt handler to run, which feeds the next available data point from the sine wave buffer to the D/A's. This has the effect of creating a continuous analog sine wave on the D/A outputs, at the sample rate and resolution defined in the program.

As with the previous examples, the linker command and Code Composer Studio make files necessary to rebuild the `WAVE` program are included with the Developer's Package. `WAVE` can serve as the basis of a waveform generator, or in concert with the A/D input capability demonstrated in `SNAP` and `TEST`, can be used to generate feedback signals in a high-speed target DSP-based servo control system.

Memory Mapping

The A4D4 module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** or the **a4d4.h** file included with the Zuma Tools DSP software library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	Read/ Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
A/D Pair 0 Data Read	R	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0
A/D Pair 0 Software Conversion	W	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0
A/D Pair 1 Data Read	R	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1
A/D Pair 1 Software Conversion	W	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1
D/A Pair 0 Data Write	W	IOMOD0 + 0x2	IOMOD4 + 0x2	IOMOD8 + 0x2
D/A Pair 1 Data Write	W	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3
D/A Pair 0 Output Update	W	IOMOD0 + 0x6	IOMOD4 + 0x6	IOMOD8 + 0x6
D/A Pair 1 Output Update	W	IOMOD0 + 0x7	IOMOD4 + 0x7	IOMOD8 + 0x7
A/D Gain Control	W	IOMOD0 + 0x8	IOMOD4 + 0x8	IOMOD8 + 0x8
A/D Pair 0 Hardware Trigger Select	W	IOMOD0 + 0xA	IOMOD4 + 0xA	IOMOD8 + 0xA
A/D Pair 1 Hardware Trigger Select	W	IOMOD0 + 0xB	IOMOD4 + 0xB	IOMOD8 + 0xB
D/A Pair 0 Hardware Trigger Select	W	IOMOD0 + 0xC	IOMOD4 + 0xC	IOMOD8 + 0xC
D/A Pair 1 Hardware Trigger Select	W	IOMOD0 + 0xD	IOMOD4 + 0xD	IOMOD8 + 0xD
IDROM SDA	R/W	IOMOD0 + 0xE	IOMOD4 + 0xE	IOMOD8 + 0xE
IDROM SCK	W	IOMOD0 + 0xF	IOMOD4 + 0xF	IOMOD8 + 0xF

TABLE 10. **A4D4 Memory Map**

Interrupt Usage

The A4D4 drives a single interrupt output on the OMNIBUS interrupt 0 pin which can be used to notify the OMNIBUS host that the current conversion has been completed. Derived from the A/D converter BUSY pin, the signal may be sourced from either A/D pair under the control of the A/D Pair 0 or Pair 1 Hardware Trigger Select Registers. Writing bit 3 true in either register selects the corresponding pair's not-busy signal as the source for the interrupt signal to the host.

Pin Connector I/O

The A4D4 output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* for additional details on how to connect to the A4D4's I/O pins.

A4D4 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50-Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1..35	34..50, 83..100	9..25, 58..75	1..25, 41..51	1..35		Reserved
36	33	8	40	36	1	IN 0+
37	82	57	39	37	9	IN 0-
38	32	7	38	38	2	IN 1+

A4D4 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50-Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
39	81	56	37	39	10	IN 1-
40	31	6	36	40	3	IN 2+
41	80	55	35	41	11	IN 2-
42	30	5	34	42	4	IN 3+
43	79	54	33	43	12	IN 3-
44	29	4	32	44	5	DAC 0
45	78	53	31	45	13	DAC 1
46	28	3	30	46	6	DAC 2
47	77	52	29	47	14	DAC 3
48	27	2	28	48	7	AGND
49	76	51	27	49	15	EXT_DAC_TRIG
50	26	1	26	50	8	EXT_ADC_TRIG

TABLE 11. A4D4 I/O Connector Pinout

Functions

Analog Input

The analog to digital converters are 16-bit devices with a 5 us maximum conversion time (Analog Devices AD976A) capable of digitizing CD quality music, sensor outputs like thermocouples and accelerometers, and other demanding acquisition applications. Each A/D channel is an independent conversion path with independent filtering and input range. The default bipolar input range is +/-10V. Custom input ranges may be achieved as described below.

The AD976A A/D converter is internally sampled prior to conversion and requires no external sample and hold circuitry. The ADC is self-timed and will typically convert in less than 5 us, which is the specified maximum conversion time for all conditions. The ADC also has its own internal precision voltage reference for accurate conversions. Do not use the reference output for any other device unless proper signal buffering is provided as this may adversely affect the conversion accuracy.

The A/D's are configured as pairs which read back with a single 32-bit read. Each 16-bit field within the readback value contains a single sample in two's-complement format. The devices may be read by a CPU read or by the DMA controllers.

Analog Input Conversion Triggering

The A/D converters may be triggered for conversion by writes to the memory-mapped A/D, externally triggered by a TTL signal, or triggered by OMNIBUS host board timers. The analog trigger selection matrix allows software to select I/O bus timer sources or external triggers for use in analog conversion triggering. The following table gives the trigger matrix control register addresses and functions for the A/D converters.

Module Register	Write Value	A/D Trigger Source Selection
A/D Pair 0 Hardware Trigger Select	0	A/D pair 0 triggered by host internal timer 0
	1	A/D pair 0 triggered by host internal timer 1
	2	A/D pair 0 triggered by host DDS timebase
	3	A/D pair 0 triggered by external A/D trigger input
	4	A/D pair 0 triggered by software access
A/D Pair 1 Hardware Trigger Select	0	A/D pair 1 triggered by host internal timer 0
	1	A/D pair 1 triggered by host internal timer 1
	2	A/D pair 1 triggered by host DDS timebase
	3	A/D pair 1 triggered by external A/D trigger input
	4	A/D pair 1 triggered by software access

TABLE 12. **A4D4 A/D Trigger Matrix Programming**

For the memory mapped conversion trigger, use a write to the A/D address. Only a write to the memory-mapped address will cause a conversion; reads are used for reading back the data.

A/D pairs conversions may also be triggered by a hardware timer or external trigger. In this case, an I/O bus host timer is programmed to run at the required sample conversion rate. The trigger selection matrix is programmed to direct the timer's output to the A/D converter pair as a conversion strobe signal. Alternatively, an external TTL trigger may be applied to the A/D external trigger input on the I/O connector.

Please note that bit 3 in the A/D Pair 0/1 Hardware Trigger Select registers enables the not-busy interrupt to the OMNIBUS host for that pair. See the Interrupts section above for details. This bit should be logically OR'd with the trigger source value from the table above if the applications requires interrupts from the A4D4.

Analog Input Trimming

The ADC offsets and gain are set at the factory, but are user trimmable with the following procedure. Errors should be trimmed at the expected normal operating temperature. The trim pot locations are given in the table below.

1. Short the differential inputs for each channel to each other and to analog ground on connector P1. You may want to do this out at your sensor to include any error sources present in the application circuitry. Trim offset pots such that a zero code is observed when continuously sampling the converters.

2. Connect one input to a voltage reference and check gain for proper codes. Full scale inputs, at a gain of 1, are +/-10V and should read raw ADC codes of 32767 and -32767 respectively for the bipolar range. Unipolar full-scale inputs are 0-10 V or 10-0 V. Trim offset pots as necessary to achieve the proper code.

Channel	Offset Trim Pot	Gain Trim Pot
0	R29	R30
1	R36	R37
2	R44	R45
3	R51	R52

TABLE 13. **A4D4 A/D Offset and Gain Adjustment Potentiometers**

Anti-alias Filtering

The A4D4 provides independent 6-pole anti-alias filtering for each A/D group. The anti-alias filters come preconfigured from the factory with a 100 kHz passband. Other filter passbands are possible through component changes in the filter circuitry. The following figures give the schematic designs for each of the input filters.

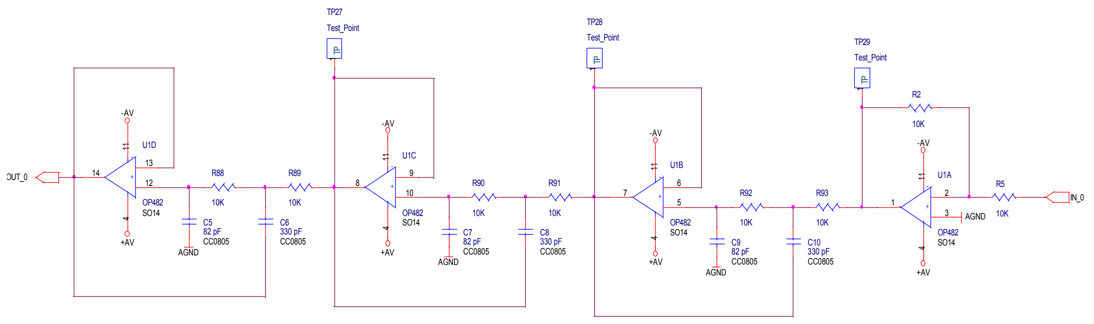


FIGURE 11. **A4D4 A/D Channel 0 Input Filter**

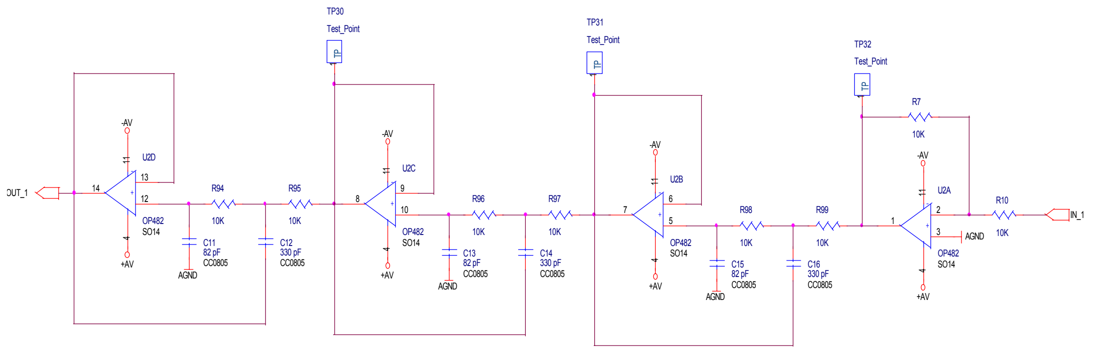


FIGURE 12. **A4D4 A/D Channel 1 Input Filter**

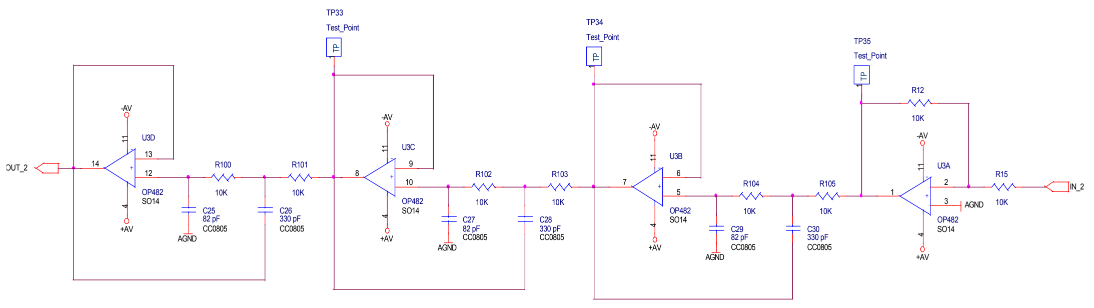


FIGURE 13. **A4D4 A/D Channel 2 Input Filter**

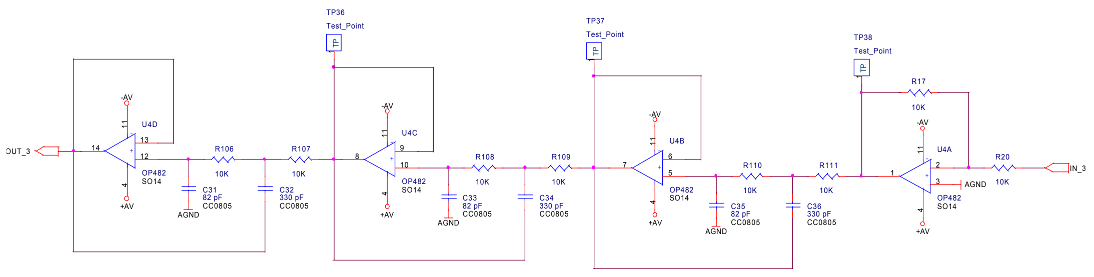


FIGURE 14. **A4D4 A/D Channel 3 Input Filter**

Programmable Gain

Each A/D has an independent software programmable gain. The standard gain selections are x1, x2, x4, or x8. An option with gains of x1, x2, x5, and x10 is also available.

The gain for a particular A/D channel is selected by writing a number in the range of 0 to 3 (indicating the gains of x1, x2, x4, and x8, respectively) to the programmable gain control register's bit field for the desired A/D channel. The following table gives the gain control register bit definition and field values.

Bit Number	**7..6**	**5..4**	**3..2**	**1..0**
Channel Affected	Ch. 3	Ch. 2	Ch. 1	Ch. 0

TABLE 14. **A4D4 Programmable Gain Control Register Bit Fields**

Bit Field Value	Gain Value (PGA206)	Gain Value (PGA207)
0	x1	x1
1	x2	x2
2	x4	x5
3	x8	x10

For example, if the application requires that channel 0 have a x2 gain, channel 1 have a x1 gain, channel 2 have a x8 gain, and channel 3 have a x4 gain. The value of the programmable gain control register should be binary 10110001 or 0xB1 hexadecimal.

Analog Output

Four channels of 16-bit instrumentation-grade digital-to-analog converters (D/A) are provided on the A4D4. The D/As are useful for analog signal outputs for both control and signal generation.

Digital data is written to the D/As for output via a set of memory mapped locations. The data bus is connected to the D/As with each pair sharing a single load location with one device occupying the lower 16 bits of the bus and the other occupying the upper 16 bits. The following table gives the load address and the bit fields used to write to each D/A device.

Module Register	Data Bus Bit Field	D/A Channel
D/A Pair 0 Data Write	D0..D15	DAC0
	D16..D31	DAC1
D/A Pair 1 Data Write	D0..D15	DAC2
	D16..D31	DAC3

TABLE 15. A4D4 D/A Channel Write Addresses and Bit Fields

Data written to the D/A devices is in straight binary format, rather than two's complement. The inverting output stage of each D/A channel yields a [+10V .. -10V] output voltage for an input code range of [0..65535].

The D/A converters are double-buffered and the A4D4 implements several update methods to accommodate various applications. Each pair of D/As shares a single update strobe line, so each D/A pair is updated simultaneously from a single trigger event. The dual D/A pairs may be updated by software triggering or by one of the hardware timebases through the use of the onboard programmable analog trigger matrix. The following section discusses triggering D/A output updates.

Analog Output Conversion Triggering

The following table gives the D/A software update addresses for each channel pair. Host CPU accesses to these locations will cause an update strobe to be driven to the corresponding D/A pair, causing the analog outputs to be updated to the current data in the D/A input latch.

Module Register	Function
D/A Pair 0 Output Update	Update D/A channel 0/1 voltage
D/A Pair 1 Output Update	Update D/A channel 2/3 voltage

TABLE 16. A4D4 D/A Software Update Control Registers

The D/A output pairs may also be updated via the hardware timer and the analog trigger selection matrix on the A4D4. In this case, an I/O bus host timer is programmed to run at the required sample conversion rate and the trigger selection matrix is programmed to direct the timer's output to the D/A converter pair as a conversion strobe signal. The trigger selection matrix works identically for the D/As as it does for the A/Ds. For a complete discussion on how to program the trigger matrix, refer to the analog input section above. The following table gives the D/A trigger matrix control register addresses and their values.

Module Register	Write Value	D/A Trigger Source Selection
D/A Pair 0 Hardware Trigger Select	0	D/A pair 0 triggered by host internal timer 0.
	1	D/A pair 0 triggered by host internal timer 1.
	2	D/A pair 0 triggered by host DDS timebase.
	3	D/A pair 0 triggered by external A/D trigger input.
	4	D/A pair 0 triggered by software access.
D/A Pair 1 Hardware Trigger Select	0	D/A pair 1 triggered by host internal timer 0.
	1	D/A pair 1 triggered by host internal timer 1.
	2	D/A pair 1 triggered by host DDS timebase.
	3	D/A pair 1 triggered by external A/D trigger input.
	4	D/A pair 1 triggered by software access.

TABLE 17. A4D4 D/A Trigger Matrix Programming

Analog Output Filtering

The D/As are amplified and filtered with high speed, low offset op amps using an inverting topology. Filtering with a simple one-pole roll-off is available using capacitors for each of the channels in parallel with the feedback resistors.

Innovative Integration recommends the use of tight tolerance (1%), low temperature coefficient resistors with low noise characteristics (such as metal film types) for these resistors. In addition, low parasitic high accuracy NPO capacitors should be used for the filter capacitors.

The following diagrams give the output filter circuit schematics for each of the D/A channels.

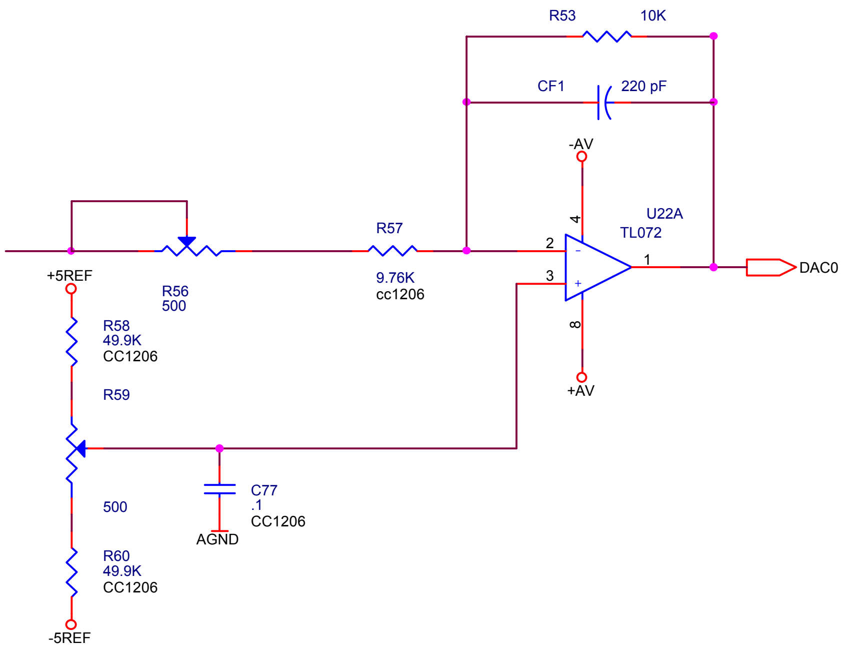


FIGURE 15. A4D4 Channel 0 D/A Output Filter

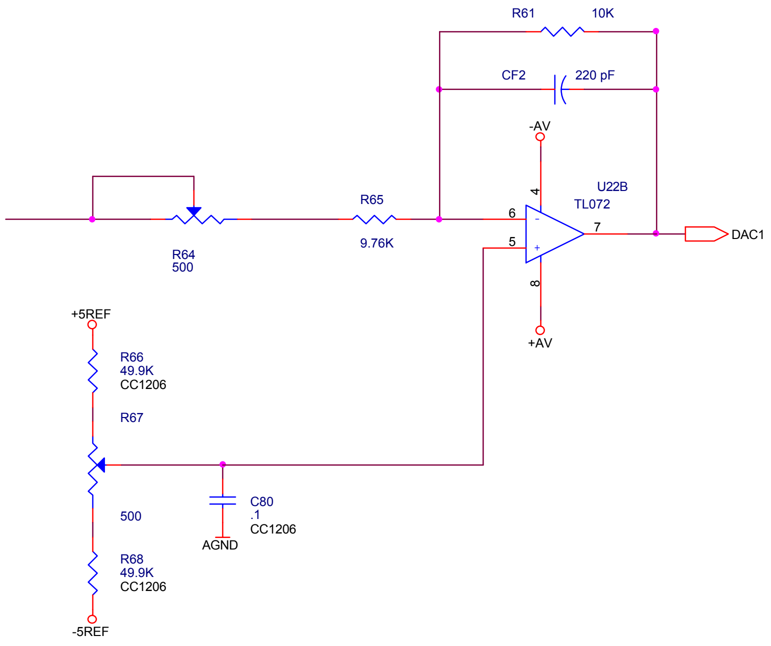


FIGURE 16. A4D4 Channel 1 D/A Output Filter

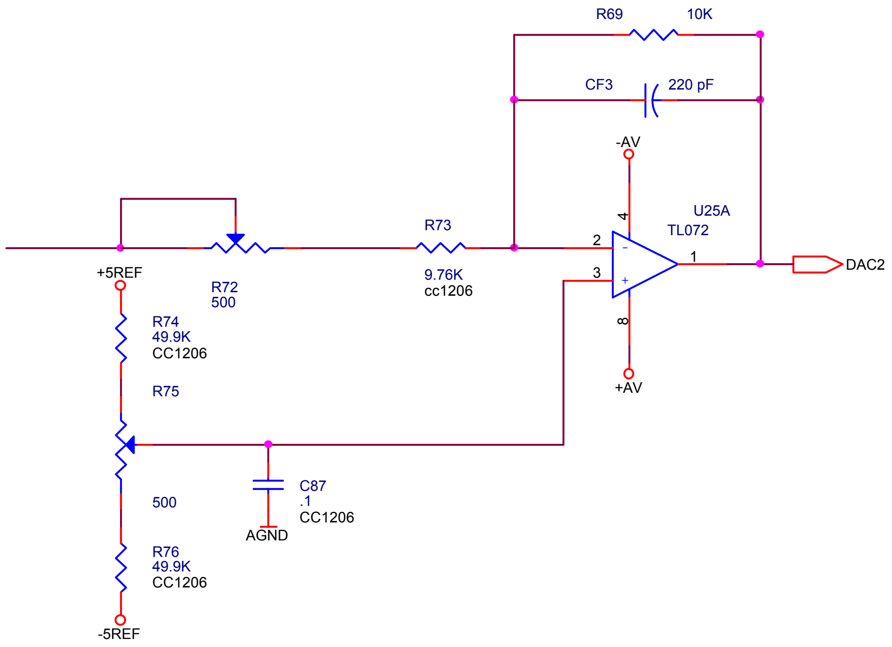


FIGURE 17. A4D4 Channel 2 D/A Output Filter

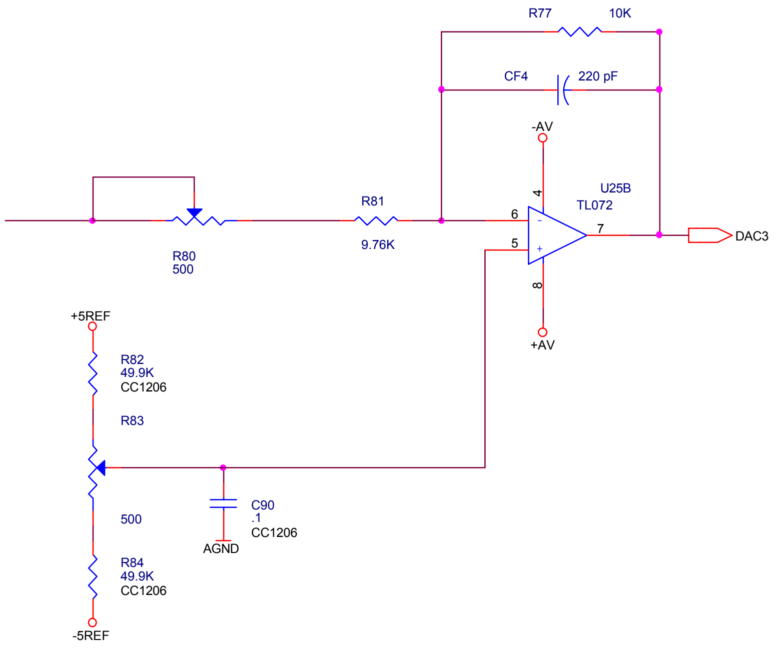


FIGURE 18. A4D4 Channel 3 D/A Output Filter

D/A Output Trimming

The A4D4 supports a trim on each D/A output for both gain and offset.

1. Write 0x8000 to the D/A channel being trimmed. Adjust the offset potentiometer to give zero volts output at the connector pin. All voltages should be measured with a short (1") ground connection to analog ground on the connector.

2. Write zero to the D/A. Adjust the gain potentiometer to give +10 volts output.

The following table gives the reference designators for the trim potentiometers.

D/A Channel	Offset Trim Pot	Gain Trim Pot
0	R59	R56
1	R66	R64
2	R75	R72
3	R83	R80

TABLE 18. A4D4 D/A Offset and Gain Adjustment Potentiometers

Some Considerations About Using the A4D4 and Term

The A4D4 module inverts the input signal going to the A/D's and then after the conversion the software multiplies the output by (-1) to correct the inverted signal. The software corrects the output with the **A4D4_correct_adc()** function located in the **adc.h** file. It should be noted that when the A4D4 and Term models are used together there is an extra level of inversion to the signal. To remove the extra level of inversion, the above mentioned function can be edited (or remove the ***A4D4_ADC_SIGN** from the equation) and then the project re-compiled. This is one way to remove the extra level of inversion if needed.

CHAPTER 4

A16D2 Module

Module Introduction

The A16D2 OMNIBUS module provides the target card processor with 16 multiplexed 16:1 channels of 200 kHz, 16-bit resolution analog input to digital output conversion (A/D). In addition, the module is equipped with two high speed, 16-bit 200 kHz digital input to analog output conversions (D/As). The A4D16 module uses one Analog Devices AD976 with independent input anti-alias filters and software programmable offset and gain amplifiers provide for flexible input. Also, a pair of Linear Technology LTC1597 16-bit, 2MHz D/As, equipped with independent output amplifiers, filtering generate clean, and high speed output signals. This module is ideal for cost-sensitive, low to mid-ranged data acquisition and control applications.

The A16D2 module allows for two different selections of input channels. They are either sixteen single-ended input channels or an optional eight differential input channels of instrumentation grade 16-bit successive approximation analog I/O. Sampling rates are supported from DC-200 kHz on a single channel or up to 12.5 kHz per channel when multiplexing through all 16 available channels.

The converters may be triggered via hardware timer or software access and are capable of interrupting the target processor in interrupt driven applications. A build auto-cal circuitry allows for in-circuit calibration to be completely under software control and eliminate the need for disconnecting cables during calibration.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module’s chapter.

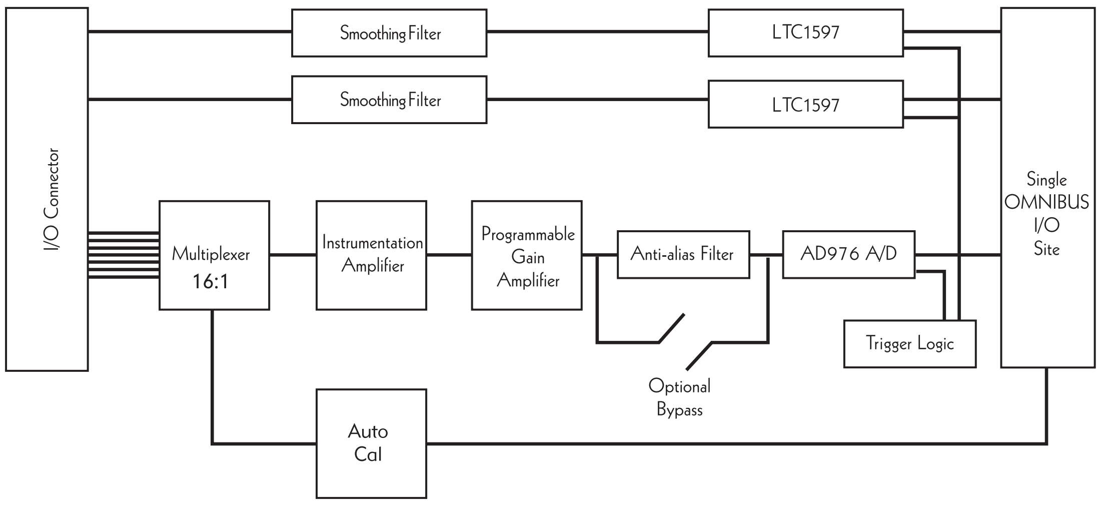


FIGURE 19. A16D2 Block Diagram

Bus Type:	Compatible with all I.I. products; 32-bit. Consumes one interrupt to host DSP. Wait-states depend on host platform.	**Filter Cutoff**	Six-pole, factory-set cutoff frequency, disabled via jumpers.
Physicals:	OMNIBUS mezzanine card; 2.0' X 4.6"	**Conversion Trigger Sources**	Software Programable, internal or external.
A/D Converters:	AD976	**Input Impedance**	1 Mohm | | 3pF
Resolution:	16-bit	**D/A Converter:**	Two LTC1597
Update Rate:	200 kHz	**Resolution:**	16-bit
Analog Input Range:	+/- 10V, +/- 5V, +/- 2.5V, +/- 1.25V software programmable	**Analog Output Range:**	+/- 10V, +/- 5V, unipolar ranges also available, set via jumper.
S/N Ratio:	83 dB	**Settling Time:**	6 us max. to 0.006% full-scale.
THD:	-70 dB (improved if filter defeated)	**Offset Error:**	Trimmable - Auto calibration
Gain Error:	Auto-Calibrated	**Gain Error:**	Trimmable - Auto calibration
Offset Error:	Auto-Calibrated	**Filter Cutoff:**	Set via capacitor.
Conversion Time	5 us	**Conversion Trigger**	Software programmable, internal or external

The target Peripheral Library provides support for each of the A4D2 functions in multiplexed and non-multiplexed, and the following sections give descriptions of the support routines. Refer to the **a16d2adc.h**, **a16d2dac.h** or **a16d2gain.h** files located in the **`c:/<target board>/Include/Target/Analog`** director and **Omnibus.hlp or Zuma.hlp** Windows helpfile for complete details on each function.

Function Name	**Operation**
A16D2_busy()	Return state of A16D2 A/D busy pin.
A16D2_convert_adc()	Manually trigger a conversion on the A/D device.
A16D2_convert_dac()	Manually trigger a conversion on selected D/A device.
A16D2_correct_adc()	Apply module-specific gain/offset corrections.
A16D2_correct_dac()	Apply module-specific gain/offset corrections.
A16D2_initialize()	Initialize A16D2 module to reset state.
A16D2_interrupt()	Set interrupt source.
A16D2_read_adc()	Read data from ADC.
A16D2_read_idrom()	Read A16D2 Module IDROM contents to buffer.
A16D2_set_adc_bipolar()	Set A16D2 A/D input mode.
A16D2_set_dac_bipolar()	Set A16D2 D/A output mode.
A16D2_set_dac_output()	Enable/disable A16D2 D/A outputs.
A16D2_set_gain()	Update gain setting for analog input.
A16D2_set_mux()	Update mux channel setting for analog input.
A16D2_trigger_adc()	Set triggering mode for ADC.
A16D2_trigger_dac()	Set triggering mode for DAC.
A16D2_write_dac()	Write value to a DAC.
A16D2_write_idrom()	Write A16D2 ID ROM from buffer.
A16D2_set_inplus()	Select source of IN+.
A16D2_set_inminus()	Select source of IN-.
A16D2_set_filter_enabled()	Control anti-alias filter enabling.
A16D2_set_mux_enabled()	Control mux output to A/D.
A16D2_AdjustGainPlot()	Adjust the specified gain pot.
A16D2_AdjustOffsetPot()	Adjust the specified offset pot.

TABLE 19. C Language A16D2 Functions

The A16D2 library functions can be divided into several groups:

1. Sample rate control.
2. A/D control.
3. D/A control.
4. Interrupt selection.
5. Identification readback.

Sample Rate Control

The sample rate to the analog hardware onboard the A16D2 may be precisely controlled via the baseboard's 9850 DDS timer. The **timebase()** function controls the sampling rate used by the A/D converters and the D/A converters within the application. Alternately either of the baseboard PIT timer channels (PIT0 or PIT1) may be used to control the sampling rate. To achieve synchronization with external events, the A16D2 can be configured to receive conversion clocks from an EXTERNAL source. The adc and dac's can be independently set by **A16D2_trigger_adc** and **A16D2_trigger_dac** functions respectively

A/D Control

The Peripheral Library includes functions for transferring data from the A/Ds, triggering A/D conversions in software, setting up a hardware timer-based trigger source, and selecting the gain and multiplexer settings on each channel.

A16D2_read_adc() is used to retrieve sample results from the A/D device. When this function is called, the current output buffer contents are returned. If a conversion is in progress, (i.e. a conversion request has been issued to the A/D, either by a software access or by a hardware timer) then the value returned is the last conversion result. The A16D2 architecture decodes a single A/D device on the low-order 16-bits of the 32-bit data bus.

The A16D2 supports conversion triggers initiated by either software accesses to the A/D conversion trigger address, or by hardware timebases. A specific software access to the module will cause a conversion to start immediately on the A/D device, while hardware triggered conversions are started by a regular timebase "tick" generated at a programmed rate from one of the target timebases. The **A16D2_convert_adc()** routine causes a software timebase conversion, while the **A16D2_trigger_adc()** function programs the A16D2 module's trigger selection matrix to select an target timebase output to trigger the desired A/D pair. Once selected, the hardware timebase must be programmed for the appropriate trigger rate by the **timebase()** function.

The gain and multiplexer control is available with the **A16D2_set_gain()** and the **A16D2_set_mux()** functions, respectively. These functions select the gain and multiplexer settings for the specified channel. To maximize channel settling time after gain and multiplexer switching, software should set the gain and input channel immediately after a conversion is triggered. This takes advantage of the A/D converter's sample and hold circuitry, which allows the A/D to take an analog sample immediately after a conversion request which is used to derive the digital conversion value. Once the sample and hold has triggered (within ~40 ns after the conversion request), the input signal may be changed without affecting the A/D's conversion result.

The module employs additional multiplexor plus sample and hold capability is available which allows up to 16 input channels to be connected to the A/D on the A16D2. The **A16D2_set_mux()** function is provided to set the mux channel selection for the A/D device.

The correct sequence to mux through all channels is as follows:

1. call **A16D2_read_adc()** to read the results of the previous conversion command. Store this into a queue or other application structure.

2. If using software-based conversions, call **A16D2_convert_adc()** to begin an A/D conversion on the new signal. For timer-based conversions, skip this step.

3. call **A16D2_set_mux()** to set the input channel.

4. call **A16D2_set_gain()** to set the channel gain.

Hardware-triggered sampling has the advantages that it automatically triggers the A/D conversion and no software access is required to the A/D. In addition, it does not require software to wait for the multiplexer to settle to each new input voltage before initiating a sample (as does software-triggered conversion).

D/A Control

The Peripheral Library includes functions for transferring data to the D/As, triggering D/A conversions in software, and setting up a hardware time-based trigger source.

A16D2_write_dac() function is used to send conversion data to one of the D/A devices. The data is latched in the D/A's input buffer, but is not reflected on the analog output of the D/A until a conversion strobe is sent to the device.

As with the A/D converters, the A16D2 supports conversion triggers to each pair of D/As initiated by either software accesses to the D/A conversion trigger address, or by hardware timebases. There are two D/A conversion trigger signals on each A16D2 module, with each signal tied to a D/A converter (thus, conversions may be initiated on individual D/A devices). Software accesses will cause a conversion to start immediately on the selected D/A device, while hardware triggered conversions are started by a regular timebase "tick" generated at a programmed rate from one of the target timebases. The **A16D2_convert_dac()** routine causes a software timebase conversion, while the **A16D2_trigger_dac()** function programs the module's trigger selection matrix to select an target timebase output to trigger the desired D/A pair. Once selected, the hardware timebase must be programmed for the appropriate trigger rate by the **timebase()** function.

If software-based D/A conversions are being performed, the correct sequence is as follows:

1. call **A16D2_write_dac()** to write the new D/A values to the required channel.

2. call **A16D2_convert_dac()** to begin a D/A conversion and update the output voltage.

If hardware based triggers are used, software only needs to load the D/A registers with the new value prior to a conversion trigger (usually done in an interrupt routine based on the trigger timebase). The conversion will be triggered by the hardware timebase.

Identification Readback

The **A16D2_read_idrom()** function is used to read the identification ROM on the module to check its identity and revision level. The function fills out an `A16D2_ID` structure with the information stored in the ID ROM on the module. The `A16D2_ID` structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string "A16D2")
2. Module revision level
3. Checksum

This data is preprogrammed at the factory and should not be altered by the user.

Example Programs for the A16D2 Module

The following sections describe the example programs available in the Developer's Package for use with a target DSP card with an A16D2 analog interface module installed.

SNAP. `SNAP` utilizes the target DSP's A16D2 module analog input circuitry to sample an external signal and calculate statistics on the resulting digital data. This program is an example of how to handle interrupt driven A/D sampling at high rates, and can serve as a test program for determining the statistical noise performance of a single A/D channel.

`SNAP` uses the target DSP trigger matrix electronics to set up an external timer to trigger conversions on a selected A/D converter channel. As conversions are triggered, the A/D's conversion complete interrupt causes an external interrupt on the target processor. This causes the interrupt handler to run, which retrieves the newly converted data and stores it to a buffer in processor memory. Once the buffer is full, the interrupt is deactivated and the program code proceeds to calculate the statistics variables on the gathered data.

For Codewright Users: For Codewright Users, the linker command (`.CMD`) files and Codewright project (`.PJT`) files necessary to rebuild the `SNAP` program are included with the Developer's Package.

For Code Composer Studio Users: For Code Composer Studio User, the make file (`.MAK`) file necessary to rebuild the `SNAP` program are included with the Developer's Package.

`SNAP` may be used as a basis for a custom data acquisition program, handling one or more A/D channels and either buffering the required data or passing the data to a host program via the card's bus mastering capability (see the section below for more information about host applications).

WAVE. `WAVE` is very similar in organization to the `SNAP` program, discussed above, and serves as an example of using the D/A converter outputs to generate signals based on digital signal data buffered in the DSP's memory.

The program first uses the C compiler's trigonometric functions to generate a scaled sine wave table in memory, in a format suitable for use by the D/A devices. The program sets up an external timer as a timebase for D/A conversions, and initializes the target DSP's trigger matrix to direct the timebase output to the D/A devices. This causes the D/A's to continuously convert the values present in their data latches. Simultaneously, the conversion trigger also causes an analog interrupt handler to run, which feeds the next available data point from the sine wave buffer to the D/A's. This has the effect of creating a continuous analog sine wave on the D/A outputs, at the sample rate and resolution defined in the program.

As with the previous examples, the linker command and Code Composer Studio make files necessary to rebuild the `WAVE` program are included with the Developer's Package. `WAVE` can serve as the basis of a waveform generator, or in concert with the A/D input capability demonstrated in `SNAP` and `TEST`, can be used to generate feedback signals in a high-speed target DSP-based servo control system.

Memory Mapping

The A16D2 module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** or the **a16d2.h** file included with the host board Development Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	**Read / Write**	**OMNIBUS Slot 0 Address**	**OMNIBUS Slot 1 Address**	**OMNIBUS Slot 2 Address**	**Bit Field Value**
A/D Data Read	R	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0	
A/D Software Conversion	W	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0	
Mux and Gain Selection NOTE: For 16 single-ended input channels, use all 16 Mux Channels. (0 to 15) For 8 differential input channels, use ONLY the first 8 Mux Channels. (0 to 7)	W	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1	Select Mux Channels use Bits 3 to 0: 0000 - Mux Channel 0 0001 - Mux Channel 1 0010 - Mux Channel 2 0011 - Mux Channel 3 0100 - Mux Channel 4 0101 - Mux Channel 5 0110 - Mux Channel 6 0111 - Mux Channel 7 1000 - Mux Channel 8 1001 - Mux Channel 9 1010 - Mux Channel 10 1011 - Mux Channel 11 1100 - Mux Channel 12 1101 - Mux Channel 13 1110 - Mux Channel 14 1111 - Mux Channel 15 Select Gain use Bits 5 to 4: 00 - Gain 0 01 - Gain 1 10 - Gain 2 11 - Gain 3

Function	Read / Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address	Bit Field Value
D/A Channel 0	R/W	IOMOD0 + 0x2	IOMOD4 + 0x2	IOMOD8 + 0x2	Read - Convert Data Write - Load Data
D/A Channel 1	R/W	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3	Read - Convert Data Write - Load Data
D/A Clock Trigger Selection	W	IOMOD0 + 0x4	IOMOD4 + 0x3	IOMOD8 + 0x3	D/A Channel 0 uses Bits 1 to 0: 00 - Timer 0 01 - Synth Clock 10 - External D/A Trigger 11 - Software Access D/A Channel 1 uses Bits 3 to 2: 00 - Timer 0 01 - Synth Clock 10 - External D/A Trigger 11 - Software Access
A/D Clock Trigger Selection	W	IOMOD0 + 0x5	IOMOD4 + 0x5	IOMOD8 + 0x5	A/D Channels use Bits 1 to 0: 00 - Timer 0 01 - Synth Clock 10 - External A/D Trigger 11 - Software Access
A/D Busy Readback	R	IOMOD0 + 0x6	IOMOD4 + 0x6	IOMOD8 + 0x6	Bit 0: 0 - Busy 1 - Available
Interrupt Control	W	IOMOD0 + 0x7	IOMOD4 + 0x7	IOMOD8 + 0x7	Enable **either** Bit 1 or 0. Bit 0 (Busy): 0 - Disable Interrupt 1 - Enable Interrupt Bit 1 (Conversion): 0 - Disable Interrupt 1 - Enable Interrupt
A/D Testing Control	W	IOMOD0+ 0x9	IOMOD4+ 0x9	IOMOD8+ 0x9	Enable **either** Bit 1 or 0: Bit 0 for +IN: 0 - Disable 1 - Enable -5V Ref Bit 1: 0 - Disable 1 - Enable AGND Enable **either** Bit 3 or 2: Bit 2 for -IN: 0 - Disable 1 - Enable -5V Ref Bit 3: 0 - Disable 1 - Enable AGND

Function	Read / Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address	Bit Field Value
A/D Range Offset and Filter Bypass	W	IOMOD0 + 0xA	IOMOD4 + 0xA	IOMOD8 + 0xA	Bit 0: **(Range Offset)** 0 - Set to -5V Ref 1 - Set to AGND Bit 1: **(Filter)** 0 - Filter Bypass 1 - Filter is ON Bit 2: **(DAC Channels)** 0 - Normal Operation 1 - Reset Bit 3: **(Mux Channels)** 0 - Disable 1 - Enable
A/D Pot Controls for Gain and Offset	W	IOMOD0 + 0xB	IOMOD4 + 0xB	IOMOD8 + 0xB	Bit 0: **(Gain)** 0 - Enable Chip Select 1 - Disable Chip Select Bit 1: **(Gain)** 0 - Enable Clock 1 - Disable Clock Bit 2: **(Offset)** 0 - Enable Chip Select 1 - Disable Chip Select Bit 3: **(Offset)** 0 - Enable Clock 1 - Disable Clock Bit 4: **(Gain and Offset)** 0 - Decrement 1 - Increment
D/As Polarity Selection	W	IOMOD0 + 0xC	IOMOD4 + 0xC	IOMOD8 + 0xC	**Unipolar (0V to +10V)** **Bipolar (-10V to +10V)** Bit 0: **(D/A Channel 0)** 0 - Unipolar 1 - Bipolar Bit 1 : **(D/A Channel 1)** 0 - Unipolar 1 - Bipolar
IDROM SDA	R/W	IOMOD3 + 0x0	IOMOD7 + 0x0	IOMOD11 + 0x0	
IDROM SCK	W	IOMOD3 + 0x1	IOMOD7 + 0x1	IOMOD11 + 0x1	
LOGIC VERSION	R	IOMOD3 + 0x3	IOMOD7 + 0x3	IOMOD11 + 0x3	

TABLE 20. **A16D2 Memory Map**

Interrupt Usage

The A16D2 drives a single interrupt output on the OMNIBUS interrupt 0 pin which can be used to notify the OMNIBUS host that the current conversion has been completed. This signal is derived from the A/D converter either on the BUSY pin or after conversion is initiated. The advantage of wiring the interrupt from conversion started is that it allows maximum time for interrupt servicing and channel changing. Writing bit 1 TRUE in Interrupt Control register select the corresponding not-busy signal as the source for the interrupt signal to the host.

Pin Connector I/O

The A16D2 output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* for additional details on how to connect to the A16D2 I/O pins.

The I/O connector pinout for 16 single-ended input channels is shown below:

A16D2 Omnibus M173280-3 Connector Pin Number	**ChicoPlus & Hombre Baseboard 100 Pin MDR Connector**		**cM44 & cM6x Baseboard SCSI-2 50-Pin Connector**	**M44, M6x, & SBC6x Baseboard IDC50 Connector**	**M44 & M6x Baseboard DB15 Connector**	**Function**
	Module Site 0	**Module Site 1**				
1..28	37..50, 87..100	12..25, 62..75	1..25, 48..50	1..28		Reserved
29	86	61	47	29		External D/A Trigger
30	36	11	46	30		Reserved
31	85	60	45	31		A/D Channel 15 Input
32	35	10	44	32		A/D Channel 14 Input
33	84	59	43	33		A/D Channel 13 Input
34	34	9	42	34		A/D Channel 12 Input
35	83	58	41	35		A/D Channel 11 Input
36	33	8	40	36	1	A/D Channel 10 Input
37	82	57	39	37	9	A/D Channel 9 Input
38	32	7	38	38	2	A/D Channel 8 Input
39	81	56	37	39	10	A/D Channel 7 Input
40	31	6	36	40	3	A/D Channel 6 Input
41	80	55	35	41	11	A/D Channel 5 Input
42	30	5	34	42	4	A/D Channel 4 Input
43	79	54	33	43	12	A/D Channel 3 Input
44	29	4	32	44	5	A/D Channel 2 Input
45	78	53	31	45	13	A/D Channel 1 Input
46	28	3	30	46	6	A/D Channel 0 Input
47	77	52	29	47	14	D/C Channel 1 Output
48	27	2	28	48	7	D/C Channel 0 Output
49	76	51	27	49	15	Analog Ground
50	26	1	26	50	8	External A/D Trigger

TABLE 21. A16D2 I/O Connector Pinout for 16 Single-ended Input Channels

The I/O connector pinout for 8 differential channels is shown below:

A16D2 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50-Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1..28	37..50, 87..100	12..25, 62..75	1-25, 48..50	1..28		Reserved
29	86	61	47	29		External D/A Trigger
30	36	11	46	30		Reserved
31	85	60	45	31		A/D Channel 7 Input -
32	35	10	44	32		A/D Channel 6 Input -
33	84	59	43	33		A/D Channel 5 Input -
34	34	9	42	34		A/D Channel 4 Input -
35	83	58	41	35		A/D Channel 3 Input -
36	33	8	40	36	1	A/D Channel 2 Input -
37	82	57	39	37	9	A/D Channel 1 Input -
38	32	7	38	38	2	A/D Channel 0 Input -
39	81	56	37	39	10	A/D Channel 7 Input +
40	31	6	36	40	3	A/D Channel 6 Input +
41	80	55	35	41	11	A/D Channel 5 Input +
42	30	5	34	42	4	A/D Channel 4 Input +
43	79	54	33	43	12	A/D Channel 3 Input +
44	29	4	32	44	5	A/D Channel 2 Input +
45	78	53	31	45	13	A/D Channel 1 Input +
46	28	3	30	46	6	A/D Channel 0 Input +
47	77	52	29	47	14	D/A Channel 1 Output
48	27	2	28	48	7	D/A Channel 0 Output
49	76	51	27	49	15	Analog Ground
50	26	1	26	50	8	External A/D Trigger

TABLE 22. **A16D2 I/O Connector Pinout for 8 Differential Input Channels**

Functions

Analog Input

The analog to digital converter is a 16-bit device with a 5 us maximum conversion time (Analog Devices AD976) capable of digitizing CD quality music, sensor outputs like thermocouples and accelerometers, and other demanding acquisition applications. The A/D channel is multiplexed into a single analog conditioning channel before reading the A/D converter. The default bipolar input range is +/-10V. Software control allows the user to select +/-10V, +/-5V, +/-2.5V, +/-1.25V, 0 to 10V, 0 to 5V, and 0 to 2.5V ranges. Custom input ranges may be achieved as described below.

The A/D converter is internally sampled prior to beginning a conversion and requires no external sample and hold circuitry. The A/D is self-timed and will typically convert in less than 5 us, which is the specified maximum conversion time for all conditions. The A/D also has its own internal precision voltage reference for accurate conversions. Do not use the reference output for any other device unless proper signal buffering is provided as this may adversely affect conversion accuracy.

The A/D is configured to read back a single 16-bit data. The 16-bit field within the readback value contains a single sample in two's-complement format. The A/D may be read by a CPU read or by the DMA controllers.

Analog Input Conversion Triggering

The A/D converters may be triggered for conversion by writes to the memory-mapped A/D, triggered by OMNIBUS host board timer, triggered by synth clock, externally triggered by a TTL signal, or triggered by software access. The analog trigger selection matrix allows software to select I/O bus timer sources or external triggers for use in analog conversion triggering. The following table gives the trigger matrix control register addresses and functions for the A/D converters.

Module Register	**Bit Field Value**	**A/D Trigger Source Selection**
A/D Clock Trigger Select	0x0	A/D triggered by host internal Timer 0.
	0x1	A/D triggered by synth clock.
	0x2	A/D triggered by external A/D trigger input.
	0x3	A/D triggered by software access.

For the memory mapped conversion trigger, use a WRITE to the A/D memory mapped address. Only a WRITE to the memory-mapped address will cause a conversion; READs are used for reading back the data.

The A/D conversions may also be triggered by a hardware timer, external trigger, or software access. In this case, an I/O bus host timer is programmed to run at the required sample conversion rate and the trigger selection matrix is programmed to direct the timer's output to the A/D converter as a conversion strobe signal. An external TTL trigger may be applied to the A/D external trigger input on the I/O connector. The software access trigger may be applied to the A/D converter with a WRITE to the memory-mapped address.

Please note that the value three (3) in the A/D Clock Trigger Select register enables the software access either for an A/D conversion or to data read. It allows the software to trigger the A/D conversion or reads the data one at a time.

Analog Input Trimming

The A/D offset and gain are set at the factory, but are user trimmable through the software program with the following programmable offset procedure. This calibration is fully software controlled, and may be performed at any time, while the system is still connected. The A/D has independent software programmable offset and errors should be trimmed at the expected normal operating temperature.

Programmable Offsets Procedure

1. Disable the Mux Channels. (See Memory Map Table, "A/D Range Offset and Filter Bypass")

2. Enable the Filter. (See Memory Map Table, "A/D Range Offset and Filter Bypass")

3. Set the Range to ground. (See Memory Map Table, “A/D Range Offset and Filter Bypass”)

4. Set both +IN and -IN to ground. (See Memory Map Table, “A/D Testing Control”)

5. Adjust the Offset Pot until A/D counts = 0 +/- 2.

For example, the programming is applied to the application for Slot 1:

A WRITE to the address: IOMOD0 + 0xA, with data: 0x3 for steps 1 to 3.

A WRITE to the address: IOMOD0 + 0x9, with data: 0xB in step 4.

A WRITE to the address: IOMOD0 + 0xB, with data: (depends on the values readback from the module) in step 5. If the Offset Pot is required to set the count upward for one time, data should be: 0x15. Then, a WRITE to the address: IOMOD0 + 0xB, with data: 0x1D to finish step 5.

Repeat step 5 until the A/D counts readback a 0 +/- 2 counts. (The average count for step 5 is 100 times)

Anti-alias Filtering

The A16D2 provides a 6-pole analog anti-alias filter for the A/D. The anti-alias filter comes preconfigured from the factory with a 100 kHz passband. Other filter passbands are possible through component changes in the filter circuitry, which may be special ordered. The following figures give the schematic designs for the input filters.

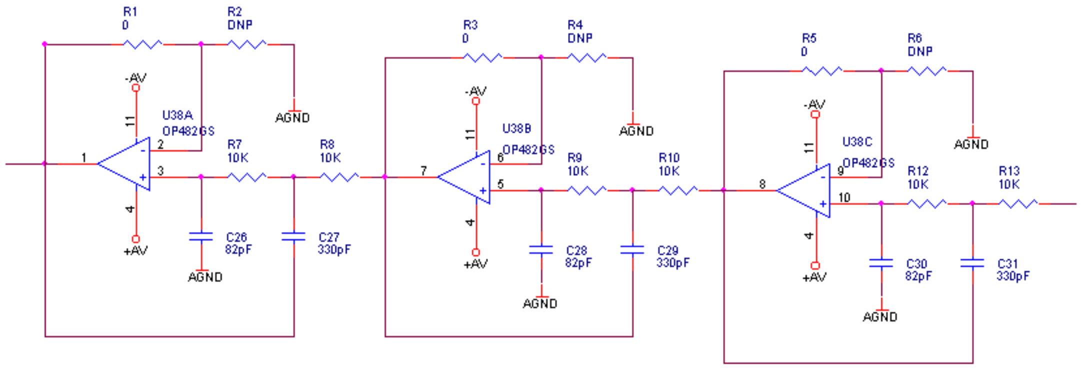


FIGURE 20. A16D2 A/D Input Filter

Two test points are located at the input and output of the filter. TP1 is the test point where it is connected to the output of the filter. TP3 is the test point where it is connected to the input of the filter.

Programmable Gain

The A/D has an independent software programmable gain. The set of standard gain selections are x1, x2, x4, or x8.

The gain for a particular A/D channel is selected by writing a number in the range of 0 to 3 (indicating the gains of x1, x2, x4, and x8, respectively) to the programmable gain control register's bit field for the desired A/D channel. The following table gives the gain control register bit definition and field values.

Address Field Value	Bit Field Value	Gain Value (PGA206)
0x1	0x0	x1
0x1	0x1	x2
0x1	0x2	x4
0x1	0x3	x8

TABLE 23. **A16D2 Programmable Gain Control Register Bit Field Values**

Analog Output

Two independent channels of 16-bit instrumentation-grade digital-to-analog converters (D/A) are provided on the A16D2. The D/As are useful for analog signal outputs for both control and signal generation.

Digital data is written to the D/As for output via a set of memory mapped locations. The data bus is connected to the D/As with sharing a single 16 bits of the bus. The following table gives the load address and the bit fields used to write to each D/A device

.

Module Register	Address Field	Value Field
D/A Channel 0	0x2	Bits [15..0]
D/A Channel 1	0x3	Bits [15..0]

TABLE 24. **A16D2 D/A Outputs Control Register**

Data written to the D/A devices is in straight binary format, rather than two's complement. The inverting output stage of each D/A channel yields a [+10V .. -10V] output voltage for an input code range of [0..65536].

The D/A converters are double-buffered and the A16D2 implements several update methods to accommodate various applications. Both D/As share a single reset line, but have separate update strobe lines. The D/As may be updated by software triggering or by one of the hardware timebases through the use of the onboard programmable analog trigger matrix. The following section discusses triggering D/A output updates.

Resetting the D/As, either at power-up or under software control, will cause the outputs to go to 0 volts.

Analog Output Conversion Triggering

The following table gives the D/A software update addresses for each of the channels. Host CPU accesses to these locations will cause an update strobe to be driven to the corresponding D/A, causing the analog outputs to be updated to the current data in the D/A input latch.

.

Module Register Function	Address	Read / Write
D/A Channel 0 Load Data	0x2	Write
D/A Channel 0 Convert Data	0x2	Read
D/A Channel 1 Load Data	0x3	Write
D/A Channel 1 Convert Data	0x3	Read

TABLE 25. A16D2 D/A Channels Addresses and Operations

The D/A outputs may be updated via hardware timer, the analog trigger, or software access of the selection matrix on the A16D2. In this case, an I/O bus host timer is programmed to run at the required sample conversion rate. The trigger selection matrix is programmed to direct the timer's output to the D/A converters as a conversion strobe signal. The trigger selection matrix works identically for the D/As as it does for the A/Ds: for a discussion on programming the trigger matrix, see the analog input section above. The following table gives the D/A trigger matrix control register addresses and values.

Module Register	Bit Field Value	D/A Trigger Source Selection
D/A Channel 0 Clock Trigger Select	Bit[1..0] = 0x0	Triggered by host internal Timer 0 as default.
	Bit[1..0] = 0x1	Triggered by host internal Synth Clock.
	Bit[1..0] = 0x2	Triggered by external A/D trigger input.
	Bit[1..0] = 0x3	Triggered by software.
D/A Channel 1 Clock Trigger Select	Bit[3..2] = 0x0	Triggered by host internal Timer 0.
	Bit[3..2] = 0x1	Triggered by host internal Synth Clock.
	Bit[3..2] = 0x2	Triggered by external A/D trigger input.
	Bit[3..2] = 0x3	Triggered by software.

TABLE 26. A16D2 D/A Trigger Matrix Programming

Analog Output Filtering

The D/As are amplified and filtered with high speed, low offset op amps. Filtering with a simple one pole roll-off at 2MHz.

Customer output bandwidth may be special ordered for lower frequencies.

D/A Output Trimming

The A16D2 requires no trimming on D/A channels at all. However, the A16D2 provides software selection on the polarities on both of the D/A channels output.

The output ranges are +/-10V, and 0 to 10V. The default range is +/-10V. Software control allows the range to also be 0 to 10V. The following table gives the software selection of output ranges for both D/A channels.

Output Range	Counts From	Mid Counts	Counts To
0V to 10V	0	32768	65535
-10V to +10V	0	16384	32767

TABLE 27. **A16D2 D/A Output Range**

The procedure for selecting the D/A output range is as follows:

1. Write to address 0xC to the D/A channels for polarity selection.
2. Write the bit field value for the desired selection.

CHAPTER 5

AD16 Module

Module Introduction

The AD16 Omnibus I/O module provides the target card with 16 channels of high speed 195 kHz, 16-bit resolution analog input to digital output conversion (A/D) per module site. There are actually 16 A/D converters for simultaneous conversion on all channels. Each of the 16 input channel consists of a high precision, DC accurate Analog Devices AD7722 A/D converter with front end conditioning circuitry, which removes the need for muxes.

The A/D's are a sigma-delta analog to digital converter eliminating the need for expensive, bulky analog anti-alias filters. Each of the inputs is a high impedance ±10v differential input with independent high accuracy instrumentation amplifiers. Additional features on each channel software programmable digital offset & gain corrections, and a high accuracy, high stability reference. All channels feature software controlled calibration (external references required). Each module also carries an identification memory that holds the module name, type, and other information to allow software to properly identify and configure the module.

The AD16 module is well suited for a variety of measurement, instrumentation, and multi-channel data acquisition applications where high dynamic range (S/N > 85 dB), DC accuracy, and simultaneous channel sampling are required. The AD16 consumes a single OMNIBUS module site and is compatible with a broad selection of DSP and data acquisition products. Some typical applications may include vibration measurements, acoustic monitoring, SONAR, and audio.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

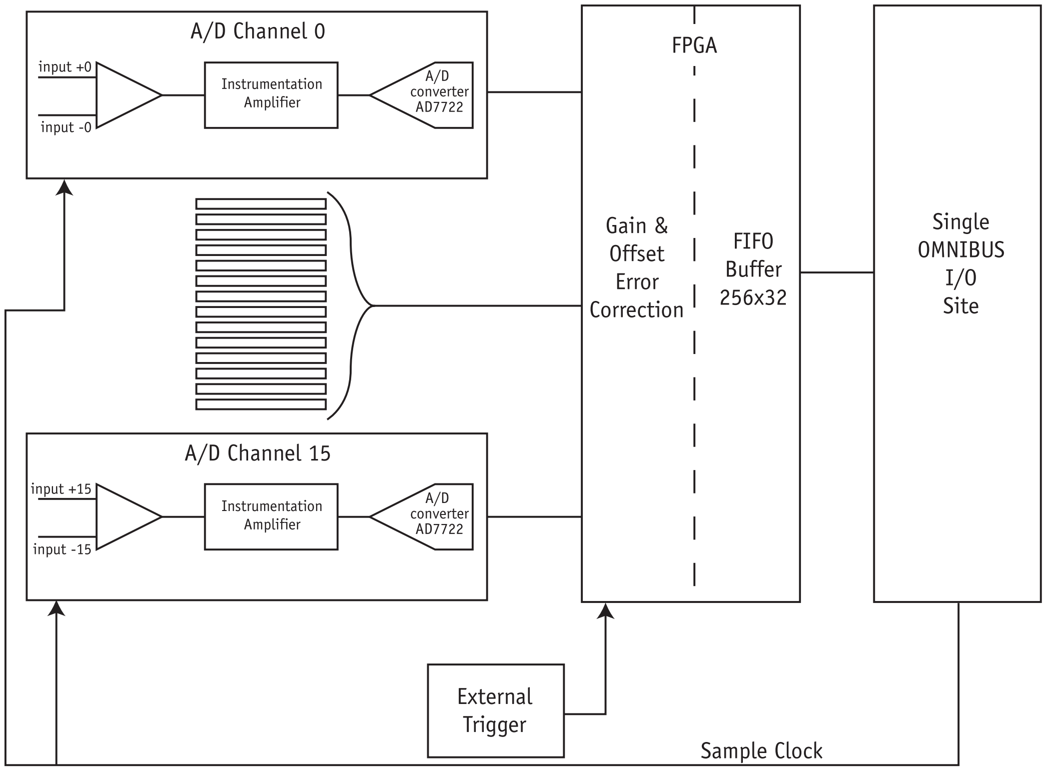


FIGURE 21. **AD16 Block Diagram**

Bus Type:	Compatible with all I.I. products; 32-bit. Consumes one interrupt to host DSP. Wait-states depend on host platform.	**A/D Converters:**	Sixteen Analog Devices AD7722 low noise and high resolution. Each converter channel has independent filtering.
Power Requirements:	3.0 W Normal (4.5 W Max.)	**Dynamic Range:**	96 dB
Physicals:	OMNIBUS mezzanine card; 2.0” X 4.6”	**Gain Error:**	Trimmable +/- 3 mV
Resolution:	16-bit	**Offset Error:**	Trimmable < 0.01% FS
Update Rate:	5 - 195 kHz.	**Input Type:**	Differential
Oversampling:	64x	**Input Impedance**	1 Mohm
Analog Input Range:	+/- 10 V differential	**Digital Filter Characteristics**	
S/N Ratio:	83 dB (98 dB Max.)	**Passbaud:**	0.496x Sample Rate
SINAD:	81 dB	**Passbaud Ripple:**	+/- 0.005 dB
ENOB:	13-bit	**Conversion Trigger Sources:**	OMNIBUS DDS Timers
SFDR:	88 dB	**Interface to DSP:**	Memory Mapped registers using FPGA interface
THD:	0.002%		

The target Peripheral Library provides support for each of the AD16 functions, and the following sections give descriptions of the support routines. Refer to the **ad16adc.h** and **ad16gain.h** files located in the `c:/<target board>/Include/Target/Analog` director and **Omnibus.hlp or Zuma.hlp** Windows helpfile for complete details on each function.

Function Name	Operation
AD16_bleed_fifo()	Read multiple A/D sample pairs from FIFO.
AD16_calibrate()	Initiate calibration cycle on all A/Ds.
AD16_convert_to_gain()	Convert float to internal fixed-point format.
AD16_empty_fifo()	Discard all FIFO contents.
AD16_enable_adc()	Specify active A/D channel pairs.
AD16_get_gain()	Adjust calibration offset for specified channel.
AD16_get_offset()	Retrieve calibration offset for specified channel.
AD16_read_adc_pair()	Read single A/D sample pair from FIFO.
AD16_read_idrom()	Read AD16 Module IDROM contents to buffer.
AD16_reset()	Reset module to power-up state.
AD16_set_fifo_threshold()	Adjust FIFO fullness level for interrupt signalling.
AD16_set_gain()	Adjust calibration gain scale factor for specified channel.
AD16_set_offset()	Retrieve calibration gain scale factor for specified channel.
AD16_start()	Begin acquiring data into FIFO.
AD16_stop()	Terminate acquiring data into FIFO.
AD16_sync()	Synchronize all A/D input channels.
AD16_write_idrom()	Write AD16 ID ROM from buffer.
AD16_fast_omnibus()	Programs Omnibus for fast FIFO accesses (For M6x only).
AD16_normal_omnibus()	Programs Omnibus for normal FIFO accesses (For M6x only).

TABLE 28. **C Language AD16 Functions**

The AD16 library functions can be divided into several groups:

1. A/D management.
2. Calibration control.
3. Interrupt selection.
4. Identification readback.

A/D Control

The Peripheral Library includes functions for transferring data from the A/D FIFO, triggering A/D conversions via a timer and setting up a hardware timer-based trigger source.

Prior to acquiring data from the AD16, the module must be initialized. Initialization consists of the following steps:

1. Reset the module to its power-up state via a call to **AD16_reset()**.
2. Initiate a calibration cycle on all A/Ds via a call to **AD16_calibrate()**.
3. Synchronize the phase of all A/D channels via a call to **AD16_sync()**.
4. Specify the fullness level of the FIFO above which the module will signal an interrupt to the baseboard via a call to **AD16_set_fifo_threshold()**.

The AD16 module utilizes a single receive FIFO into which all samples from all channels are stored during the acquisition process. Eight pairs of A/D converters are available on the module. Each pair presents 32-bit data to the FIFO. The low-order sixteen bits of the 32-bit data corresponds to the converted results from the lower-numbered A/D in the pair. The high-order 16-bits of the 32-bit data corresponds to the converted results from the higher-numbered A/D in the pair. Only the sample results from A/D pairs, which are currently enabled, are stored in the FIFO during acquisition. Individual channel pairs are enabled via the **AD16_enable_adc()** function.

During acquisition, sample pairs are added into the FIFO starting with the lowest-numbered, enabled A/D pair and ending with the highest-numbered, enabled A/D pair. Thus, the data stored in the FIFO consists of data received from *active* A/D channel pairs only. For each conversion trigger received by the module, one 32-bit value from each enabled channel pair is stored into the FIFO. The term sweep is used within this documentation to denote the collection of 32-bit sample pairs stored into the FIFO for each trigger event.

Acquisition is enabled via the **AD16_start()** function. Afterwards, 32-bit sample data for all enabled pairs (sweeps) are added to the FIFO each time the A/Ds are triggered via the conversion clock. All enabled A/Ds are always simultaneously triggered each time a trigger pulse in received by the module.

When the FIFO reaches the previously programmed fullness level, the module can assert an interrupt to the baseboard. The baseboard can be programmed to bleed data from the FIFO either using an interrupt handler or via DMA.

Within an interrupt handler, use the **AD16_read_adc_pair()** function to read the oldest channel pair result from the FIFO. Repeat this call for as many pairs as have been enabled in order to remain aligned on sweep boundaries.

Alternately, application software may call the **AD16_bleed_fifo()** function to bleed multiple channel pairs into a memory buffer using a specified DMA channel. This usually results in superior real-time performance.

Calibration

The AD16 has been architected without using trimpots for calibration of A/D scalefactor and offset. Instead, the onboard logic mathematically corrects the results of each analog input channel according to the formula $y = mx + b$ where m is the scalefactor (gain) to apply to the input channel and b is the offset to apply to the input channel. Separate gain and offset values are supported for each analog input channel.

During factory calibration, the optimal coefficients for the gain and offset for each channel have been measured and stored into the flash ROM onboard the AD16 module. These coefficients must be retrieved at runtime via the **AD16_read_idrom()** function and stored into the AD16 in order to obtain measurements within the factory-specified accuracy of the module.

The **AD16_set_gain()** and **AD16_set_offset()** functions are available for use in application software to programmatically adjust the gain and offset of all channels of the module.

Identification Readback

The **AD16_read_idrom()** function is used to read the identification ROM on the module to retrieve factory calibration coefficients, check its identity and revision level. The function fills out an `AD16_ID` structure with the information stored in the ID ROM on the module. The `AD16_ID` structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string “AD16”)
2. Module revision level
3. Calibration coefficients for gain and offset corresponding to each A/D channel
4. Checksum

This data is preprogrammed at the factory and should not be altered by the user.

Example Programs for the AD16 Module

The following sections describe the example programs available in the Developer’s Package for use with a target DSP card with an AD16 analog interface module installed.

SNAP. `SNAP` utilizes the target DSP’s AD16 module analog input circuitry to sample an external signal and calculate statistics on the resulting digital data. This program is an example of how to handle interrupt driven A/D sampling at high rates, and can serve as a test program for determining the statistical noise performance of a single A/D channel.

SNAP uses the target DSP trigger matrix electronics to set up an external timer to trigger conversions on a selected A/D converter channel. As conversions are triggered, the A/D's conversion complete interrupt causes an external interrupt on the target processor. This causes the interrupt handler to run, which retrieves the newly converted data and stores it to a buffer in processor memory. Once the buffer is full, the interrupt is deactivated and the program code proceeds to calculate the statistics variables on the gathered data.

A closely-related example, SNAP4.C, differs from the SNAP.C example above in that two A/D channel pairs are simultaneously enabled, rather than a single A/D channel pair.

For Codewright Users: For Codewright Users, the linker command (`.CMD`) files and Codewright project (`.PJT`) files necessary to rebuild the `SNAP` program are included with the Developer's Package.

For Code Composer Studio Users: For Code Composer Studio User, the make file (`.MAK`) file necessary to rebuild the `SNAP` program are included with the Developer's Package.

`SNAP` may be used as a basis for a custom data acquisition program, handling one or more A/D channels and either buffering the required data or passing the data to a host program via the card's bus mastering capability (see the section below for more information about host applications).

Memory Mapping

The AD16 module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** or the **ad16.h** file included with the host board's Development Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	Read/Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
A/D Data FIFO	R	IOMOD0 + 0	IOMOD4 + 0	IOMOD8 + 0
A/D Channel Enable Register	W	IOMOD0 + 1	IOMOD4 + 1	IOMOD8 + 1
Control Register	W	IOMOD0 + 2	IOMOD4 + 2	IOMOD8 + 2
FIFO Threshold	W	IOMOD0 + 5	IOMOD4 + 5	IOMOD8 + 5
Gain Coefficient Memory	R/W	IOMOD1 + 0..F	IOMOD5 + 0..F	IOMOD9 + 0..F
Offset Coefficient Memory	R/W	IOMOD2 + 0..F	IOMOD6 + 0..F	IOMOD10 + 0..F
IDROM	R/W	IOMOD3	IOMOD7	IOMOD11

TABLE 29. **AD16 Memory Map**

Interrupt Usage

The AD16 has a single interrupt output that indicates when data is available to be read from the FIFO. The interrupt can be programmed to trigger on the FIFO not empty condition or when the FIFO exceeds a set threshold condition. This feature allows the programmer to pace the data retrieval from the module based upon the expected data rate.

The interrupt enable and mode selection is controlled by the control register, while the threshold value (if used) is controlled by the FIFO threshold register. See below for details on programming these features.

Pin Connector I/O

The AD16 output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* for additional details on how to connect to the AD16's I/O pins.

AD16 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50 Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1	100	75	25	1		Reserved
2	50	25	24	2		Reserved
3	99	74	23	3		Reserved
4	49	24	22	4		Input 15 +
5	98	73	21	5		Input 14 +
6	48	23	20	6		Input 13 +
7	97	72	19	7		Input 12 +
8	47	22	18	8		Input 11 +
9	96	71	17	9		Input 10 +
10	46	21	16	10		Input 9 +
11	95	70	15	11		Input 8 +
12	45	20	14	12		Reserved
13	94	69	13	13		Reserved
14	44	19	12	14		Reserved
15	93	68	11	15		Input 7 +
16	43	18	10	16		Input 6 +
17	92	67	9	17		Input 5 +
18	42	17	8	18		Input 4 +
19	91	66	7	19		Input 3 +
20	41	16	6	20		Input 2 +
21	90	65	5	21		Input 1 +
22	40	15	4	22		Input 0 +
23	89	64	3	23		Reserved
24	39	14	2	24		Reserved
25	88	63	1	25		AGND
26	38	13	50	26		Reserved
27	87	62	49	27		Reserved

AD16 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50 Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
28	37	12	48	28		Reserved
29	86	61	47	29		Input 15 -
30	36	11	46	30		Input 14 -
31	85	60	45	31		Input 13 -
32	35	10	44	32		Input 12 -
33	84	59	43	33		Input 11 -
34	34	9	42	34		Input 10 -
35	83	58	41	35		Input 9 -
36	33	8	40	36	1	Input 8 -
37	82	57	39	37	9	Reserved
38	32	7	38	38	2	Reserved
39	81	56	37	39	10	Reserved
40	31	6	36	40	3	Input 7 -
41	80	55	35	41	11	Input 6 -
42	30	5	34	42	4	Input 5 -
43	79	54	33	43	12	Input 4 -
44	29	4	32	44	5	Input 3 -
45	78	53	31	45	13	Input 2 -
46	28	3	30	46	6	Input 1 -
47	77	52	29	47	14	Input 0 -
48	27	2	28	48	7	Reserved
49	76	51	27	49	15	External Gate
50	26	1	26	50	8	AGND

TABLE 30. AD16 I/O Connector Pinout

Functions

The following block diagram shows the AD16's functions.

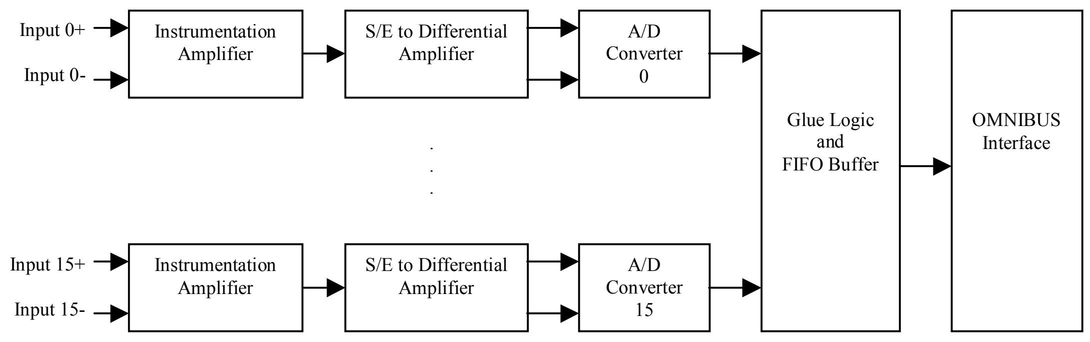

Input 0+
Input 0-
Instrumentation Amplifier
S/E to Differential Amplifier
A/D Converter 0
Glue Logic and FIFO Buffer
OMNIBUS Interface
Input 15+
Input 15-
Instrumentation Amplifier
S/E to Differential Amplifier
A/D Converter 15

FIGURE 22. AD16 Block Diagram

A/D Inputs

As shown in the block diagram above, each input channel on the AD16 consists of a set of input amplifiers, the A/D converter itself, a digital trim and FIFO buffer for recovering converted data along with interfacing this data to the OMNIBUS bus interface.

The input stage amplifier consists of an instrumentation amplifier (Linear Technology LT1176) which present a high input impedance to the input source signal and which may be used to interface to differential input signals. Nominal input range for the AD16 is +/-10V, although this may be decreased by changing the instrumentation amplifier’s gain (see below for details). No provision is made in the AD16 design for accommodating offset input signals (i.e. signals which do not swing symmetrically around ground).

The second amplifier is a differential amplifier used drive the A/D inputs. The differential amplifier has extremely good output gain matching capability so that no measurable differential errors are introduced. A single-pole, low pass filter on each A/D input is set to a nominal –3 dB roll-off point of 5.3 MHz (one-half the A/D analog sample frequency, or 64 times the maximum output word rate). Since the sigma-delta A/D samples at a very high speed, this filter will be adequate to eliminate noise at the data rates up to the 195 kHz maximum sample rate. The internal filter of the sigma-delta A/D has a nominal step response settling time of 425 us, and a group delay of 216 us for an input clock of 12.5 MHz to the A/D. These times scale along with the input clock rate.

Following conversion by the A/Ds the signal data is clocked using the A/D’s serial interface into the control logic, where a digital trim is performed on the data. The results stored to the buffer FIFO for readback by the OMNIBUS host. The following sections detail the use and programming of the digital trim feature as well as the FIFO buffer interface.

A/D Control Register

The A/D control register gives basic control over the calibration and reset of the AD7722 devices along with start and stop control over the module logic. The system may command a reset of the A/Ds, thus clearing the internal filters or may command a calibration. This control register acts on *all* A/D channels at the same time; therefore the calibration, synchronization, and reset functions must be done on all the channels in unison.

Bit Number:	31-8	7-6	5	4	3	2	1	0
Bit Field:	Reserved	INTSEL	Reserved	EXTGATE	SYNC	CAL	RESET	RUN

FIGURE 23. **AD16 A/D Control Register**

Bit Field Name	Function
RUN	Run/Stop data acquisition: 1= run, 0 = stop
RESET	A/D reset control: 1 = A/Ds in reset, 0 = A/Ds not in reset
CAL	A/D calibration control: write of 1 triggers calibration request
SYNC	A/D sync control: 1 to 0 transition triggers A/D filter synchronization
EXTGATE	Enables an external gate signal to trigger I/O: 1=run, 0=stop
INTSEL	Select interrupt type from module: 00= interrupt disabled (default after reset). 01 = interrupt on FIFO not empty. 10 = interrupt on FIFO threshold level exceeded. 11= reserved.

TABLE 31. **AD16 A/D Control Register Definition**

Reset Bit

The reset bit will clear all of the A/D converters' internal calibration settings and digital filters. Following a reset all A/Ds must be internally calibrated using the calibration control bit (see below, and see the Initialization Issues section for more information).

The reset bit is active high (1). The power-up condition is with the A/D chips in the reset mode.

Calibration Bit

The calibration bit causes each A/D to initiate an internal calibration. The A/D calculates its internal calibration coefficients and uses them to correct subsequent data from the converter. The reset and synchronization bits must both be false to initiate a calibration. Note that an internal A/D calibration does not affect the external circuitry on the AD16 module (i.e. this is not a trim for external offsets). See below for more information concerning the AD16's digital calibration features.

Synchronization Bit

The synchronization bit allows multiple A/D channels to sample simultaneously by allowing the user to initialize the A/D internal filters and start the A/D filters simultaneously. The synchronization feature should be used to start the AD16 so that all converters sample simultaneously. The calibration bit should be false when synchronizing the A/D channels using the synchronization bit.

Analog Gain Adjustment

The instrumentation amplifier front end of each A/D channel may be adjusted for gains from 1 to 1000 by adding a resistor to the input stage. The factory default setting is a gain of one (no resistor installed). The following table shows the gain adjustment resistors for each channel. The gain may be calculated as shown.

$$\text{Gain} = (49.4\text{k}\Omega/\text{Rg}) + 1$$

Which may be solved for Rg:

$$\text{Rg} = 49.4\text{k}\Omega/(\text{Gain}-1)$$

The following figure gives the typical schematic design of the instrumentation amplifier input stage.

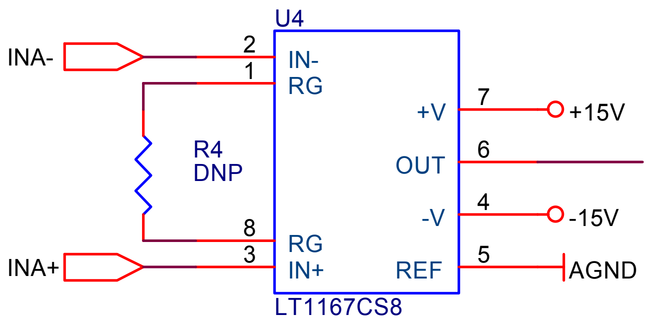


FIGURE 24. **AD16 Channel 0 Instrumentation Amplifier Front End Schematic**

The following table gives the reference designators for each input channel by channel number. All resistors are surface mount 1% metal film, size 0603.

Channel	Gain Resistor
0	R4
1	R12
2	R18
3	R24
4	R30
5	R36
6	R42
7	R48
8	R54

Channel	Gain Resistor
9	R60
10	R66
11	R72
12	R78
13	R84
14	R90
15	R96

TABLE 32. **AD16 Instrumentation Amplifier Gain Resistors**

Note that signal frequency response of the instrumentation amplifier is a function of the amplifier gain level, with higher gains resulting in reduced bandwidth. See the Linear Technology LT1176 data sheet for more information concerning the instrumentation amplifier's frequency response characteristics versus gain level.

IMPORTANT NOTE: *Do not exceed +/-15 V input relative to the analog ground under any condition or damage to the AD16 module may occur.*

Digital Trim

In addition to the internal calibration performed by the A/D converter, the AD16 also provides a digital trim feature. This feature allows incoming digital data samples from the A/D converter to be digitally trimmed for offset and gain errors before the data is stored to the FIFO buffer for retrieval by the OMNIBUS host. This trim is performed automatically in hardware with no software overhead (except initial setup).

Digital trim on each channel is based on a set of gain and offset trim coefficients stored to registers in the AD16 logic. One pair of registers is provided per input channel to allow for independent trim of all 16 channels. The trim function is defined mathematically as follows:

Corrected data = (gain * A/D) + offset

Trim arithmetic is performed using fixed point hardware and results in a two's complement fixed point output. The gain coefficients are stored by the hardware as 17-bit numbers with a binary point between bits 15 and 16. This encoding results in a range of possible gain values from 0.0 to slightly below 2.0, where binary values with the MSB (bit-16) set result in gain values equal to or above 1.0 and binary values with the MSB clear result in gain values below 1.0. The following table gives example binary codes for several gain coefficients in order to illustrate the encoding method.

Binary Gain Coefficient Value	Gain Value
0x1C000	1.75

Binary Gain Coefficient Value	Gain Value
0x18000	1.5
0x10000	1.0
0x08000	0.5
0x04000	0.25
0x00000	0.0

TABLE 33. **AD16 Digital Trim Gain Coefficient Examples**

Offset coefficients are stored as two's complement integers with a 16-bit width. Since the offset coefficients are signed, a negative offset coefficient value will cause the trim results to decrease in absolute value while a positive offset coefficient value will cause the results to increase in absolute value.

The gain and offset coefficients are stored in the gain and offset coefficient memory registers. There are 16 locations are provided for each type of coefficient.

Each module is delivered with a set of factory calibration coefficients that have been stored in the non-volatile on-module IDROM memory. Development system software for the OMNIBUS host card contains support for copying these coefficients from the IDROM to the coefficient registers.

Please note that digital trim does affect the absolute input range of the AD16, since the trim process is performed in the digital domain (i.e. after conversion from the original analog signal, as opposed to a trim performed in the analog domain prior to conversion). Onboard offset and gain errors, although typically small and trimmable by the digital trim feature, will cause a digital dynamic range loss.

For example, in the case of an input channel, which exhibited no gain error and an onboard offset error of +100 counts, the digital trim would typically be programmed to subtract 100 counts from the digitized data. This would have the effect of removing the offset error, but would result in a requirement that the input signal be limited to a range of approximately +/-9.970V in order to avoid exceeding the 16-bit digital dynamic range of the resulting trimmed data. The 30 mV of reduced amplitude is equal to 100 counts in the 16-bit digital realm, referenced to a nominal +/-10V input range (20 volts of input swing divided by 65536 counts of A/D converter dynamic range results in an analog bit resolution of approximately 300 microvolts). Since the onboard offset raises the input by 30 mV (100 counts of measured offset), the positive amplitude of +/-9.970 V input signal would just hit the maximum digital count value of the converter. But since the overall peak to peak swing of the signal is only 19.94 volts, approximately 200 counts of A/D dynamic range is unusable (i.e. the lowest converted value possible in this case is approximately –32568, instead of the theoretically lowest value of –32768).

IMPORTANT NOTE: *The AD16 digital trimming feature does not limit its output in the case of an overrange due to an excessively large absolute gain or offset coefficient or overranging due to the input signal. Digital outputs of the trim feature in such cases will swing past the normal resolution limit and wrap around through the opposite limit. For example, in the case of an input which is rising towards the positive digital rail, if the input continues to rise and the offset coefficient is set such that the resulting output data is larger than the maximum 16-bit amplitude. The data will wrap around and become a negative number with decreasing absolute amplitude.*

Please note that this affect will not occur if the digital trim is set to a gain of 1.0 and an offset of zero (power-on reset defaults), due to the fact that the converter's output data cannot exceed the 16-bit 2's complement numerical range and the trim system is set to pass the data unchanged. The effect will also not occur if the gain coefficient is set to less than 1.0 and the offset is set to zero.

Users should pay careful attention to input signal absolute ranges and be sure that all coefficients are set to match the input signal. Users should also be careful that any changes to the instrumentation amplifier gain are also taken into account.

Synchronizing Channels and Modules

Multiple AD16 modules may be synchronized by linking the synchronization signals between modules. Sync input and output signals allow software sync commands from one AD16 module to be shared with other AD16 modules on the same host. This allows a single sync command (write to the control register enabling sync active) to synchronize multiple AD16 modules and cause simultaneous sampling across the converters present on those modules.

Connector (JP2) provides access to the synchronization input and output signals. The following table gives the pinouts for (JP2).

Pin number	Function
1	Sync output
2	Sync input

TABLE 34. **AD16 Sync Connector Pinouts**

The sync output pin automatically drives out the sync bit value from the control register. When host software enables synchronization, a pulse is generated on the sync output pin with the same timing as the pulse generated to the A/D converters. This pulse, if connected to the sync input pin on a second AD16 module, will cause the converters on the second module to become synchronized with the module generating the sync pulse.

If the modules are installed on the same host card, they will have exact synchronization since they use the same A/D clock. If they are on different host cards, and the A/D clock cannot be shared *but is set to the same rate,* then they can be synchronized to within one A/D clock. Usually, this is acceptable since the A/D clock is 12.5 MHz which results in a maximum initial offset of less than 80 ns. The exact relationship may drift slightly over time as the clocks changes over time, but this usually has a small effect due to the high clock rates used. If a common clock can be sent to each card, for example using SynCLink on Chico or an external clock source, this effect may be eliminated.

FIFO Data Buffering

Data results from the digital trimming logic are stored in the AD16's FIFO data buffer for readback by the OMNIBUS host. The FIFO memory buffers data until the host is able to read it, and implements several programmable options which allow the AD16 to adapt to specific application requirements.

FIFO buffer memory on the AD16 is 256 32-bit words in length, allowing up to 512 samples to be stored as channel pair combinations (see below for more information on the FIFO storage format).

The AD16 allows the user to select the number of channel pairs of data to acquire and to select the type of interrupt notification the application should receive from the module concerning the availability of data in the FIFO.

FIFO Channel Enables

The AD16 always acquires data from all 16 A/D channels and processes the data through the digital trim logic. Although it also allows the user to select which channel's data will be saved to the FIFO in applications which do not require data from all 16 available channels. In such applications FIFO buffer memory and OMNIBUS bandwidth are conserved by transferring only the data which is required by the end user.

The FIFO channel enable register allows the user to select which pairs of data will be saved in the FIFO. Consisting of eight bits, the register allows data transmission operations to be enabled or disabled on an individual channel pair basis. The following diagram gives the channel enable register definition.

Bit Number:	31-8	7	6	5	4	3	2	1	0
Bit Field:	Reserved	Ch 14,15	Ch 12,13	Ch 10,11	Ch 8,9	Ch 6,7	Ch 4,5	Ch 2,3	Ch 0,1

FIGURE 25. **AD16 FIFO Channel Enable Register**

Enabling a subset of the channels allows the application to only read the channel pairs that are of interest to the application. This will reduce the data rate from the module by eliminating the data from unneeded channels. For example if channel pairs 0, 5, 6 (channel 0 and 1, 10 and 11, and 12 and 13) are enabled in the A/D channel enable register, the data read from the FIFO will be arranged in sets of three (3) with pairs 0, 5 and 6 making the set (we will call each set an event). The following diagram shows the storage of data in the FIFO when only a subset of the channels are enabled.

FIFO Location n*3 + 2	Pair 6	
FIFO Location n*3 + 1	Pair 5	Event n, Sampled at t = n
FIFO Location n*3	Pair 0	
...	…	...
FIFO Location 5	Pair 6	
FIFO Location 4	Pair 5	Event 1, Sampled at t = 1
FIFO Location 3	Pair 0	
FIFO Location 2	Pair 6	
FIFO Location 1	Pair 5	Event 0, Sampled at t = 0
FIFO Location 0	Pair 0	

FIGURE 26. AD16 FIFO Data Storage Example

Each event is a collection of the enabled channel pairs of data that is sampled at the same time. The total FIFO depth is 256 pair points, so the number of events that may be stored is equal to 256 divided by the number of channel pairs enabled.

AD16 Interrupts and FIFO Thresholds

Timing of OMNIBUS host data reads during AD16 data acquisition is based on a status interrupt sent from the AD16 to the host. The interrupt informs the host that a certain amount of data is available to be read from the AD16 FIFO. The interrupt may be programmed to go active when the FIFO is either not empty (at least one data point is available to be read) or has a reached a preprogrammed fullness threshold level. The interrupt type is selected by the A/D control register (see above for details): The FIFO fullness level (in the case of threshold based interrupts) is selected by the FIFO threshold control register.

The threshold value is defined as the number of points total (for all channels) which will be stored in the FIFO before the AD16 interrupt triggers a host read of FIFO data. The FIFO threshold control is an eight-bit register, allowing threshold values from 0 to 255. It is recommended that the threshold value be an integer multiple of the number of enabled channel pairs, so that data read from the FIFO represents sets of data from all enabled A/D pairs. For example, if 4 channel pairs are enabled, a FIFO setting of 32 would give 8 data sets of the 4 channel pairs, with a buffer of 224 (256-32) data points remaining available in the FIFO.

Handling of data readback triggered by FIFO threshold interrupts is straightforward. Once the data acquisition sequence has been started, data begins to fill the FIFO. Once the FIFO reaches the threshold values, an interrupt signal is sent to the host. An appropriately enabled host interrupt handler then runs and copies an amount of data equal to the threshold value from the AD16 FIFO to local storage within the host. The interrupt handler always copies the threshold amount of data from the FIFO because the

AD16 programming guarantees that at least this amount of data will be available. This condition avoids read underrun conditions where the FIFO does not have enough data to fulfill the host's requirements.

Note: it is generally a good idea to clear the host's interrupt status flags pertaining to the AD16 after finishing the FIFO copy and before exiting the interrupt handler. This is recommended due to a potential race condition, which can occur between the host's data readback and the continued FIFO fill operation being performed by the AD16. Since (in continuous data acquisition applications) the AD16 will continue filling the FIFO after triggering the host interrupt, the FIFO will typically be slightly more full than the threshold point by the time the host begins reading data. Since the host is typically much faster at emptying the FIFO than the AD16 is at filling it (a prerequisite in order to avoid FIFO overflow) the FIFO will at some point pass back through the threshold fullness level as it empties. If the AD16 adds an additional data point at the right time, the interrupt will trigger again even though the host is currently reading data and will eventually empty the FIFO back down to the nearly empty point. The extra interrupt (falsely triggered, since the FIFO ends up nearly empty) will cause a read underrun when the host tries to once again read the threshold amount of data from the FIFO. Clearing the appropriate interrupt status bit on the host processor before leaving the interrupt handler precludes the spurious interrupt from occurring.

A/D FIFO Data Output Format

A/D data is read from the AD16 FIFO as 32-bit words comprised of pairs of matched odd and even channel data. Channels are paired together for readback starting with channels 0 and 1 continuing up through channels 14 and 15 as the last pair combination. Combined data from each A/D pair is read as a single 32-bit value, with the two 16-bit A/D values from each channel in the pair each taking up half of the data bus on each read cycle. The upper half of the 32-bit word consists of the odd numbered channels (1,3,5, and so on up to 15) while the lower half consists of data recovered from the even numbered channels (0,2,4, and so on up to 14). Data is always read as matched consecutive pairs: the AD16 does not support reading data from single channels or combining channels which are not consecutive even/odd pairs (i.e. channels 0 and 1 may be read back as a pair but channels 1 and 2 may not).

The following diagram shows the bit positions of each channel in the paired readback data.

Bit Number:	31-16	15- 0
Bit Field:	Odd A/D channel data	Even A/D channel data

FIGURE 27. **AD16 A/D FIFO Data Output Format**

The output codes for the A/D are shown in the following table for the factory default +10V to –10V input range. The output data is in two's complement format.

Input Voltage	A/D Code Returned
+10 V	0x7FFF
0 V	0x0
-10 V	0x8000

TABLE 35. AD16 A/D Data Coding

Clocking the AD16

The AD16 requires a high frequency sample clock to drive the A/D converters. This clock is used by the AD7722 converters to drive the sigma-delta mechanism of the converters. This clock is provided by the host's DDS timebase, or by a clock oscillator in the case of hosts (such as Chico) which do not provide a DDS timebase generator.

The DDS signal provided by the OMNIBUS host is divided by a factor of two in the AD16 logic before being delivered to the converters as a timebase. Therefore the clock source must be programmed to a frequency equal to 128 times the resultant sample rate required by the application (the 64 times sample clock required by the A/D multiplied by two to compensate for the divider in the AD16 glue logic).

Please note that all converters run off of the same sample clock. The AD16 does not support the use of multiple sample clocks.

Even though the AD7722 is specified with a master clock rate of 12.5 MHz, the AD7722 can operate with clock frequencies up to 15 MHz and as low as 300 kHz. The input sample rate, output word rate, and the frequency response of the digital filter are directly proportional to the master clock frequency. For example, reducing the clock frequency to 5 MHz leads to an analog input sample rate of 10 MHz, an output word rate of 78.125 kSPS, a pass-band frequency of 36.25 kHz, a cutoff frequency of 38.77 kHz, and a stop band frequency of 41.875 kHz.

A/D performance is not characterized other than 12.5 MHz, and the performance may degrade at other frequencies. In addition, changing the clock rate requires re-calibration of the AD7722.

Initialization Issues

The following steps are required for proper initialization and operation of the converters on the AD16.

1. Turn on the DDS clock on the host card. This is the master clock for all A/D converters.
2. Reset the A/D converters by toggling the reset bit in the control register high (1) then low (0).
3. Turn off the sync bit.

4. Perform an A/D calibration. This will require a software delay of 1.5 milliseconds.
5. Toggle the sync bit high the low to synchronize the converters.
6. Write all digital trim values out to the gain and offset coefficient memory locations.
7. Install the host interrupt handler and enable the interrupt to the DSP.
8. Turn on the run bit.

Some Considerations About Using the AD16

Data Rates

The AD16 module has sixteen A/D channels with a capability of 192 kHz data rate per channel. Since the channels are paired on the bus (two channels are read for each 32 bit access), this results in a maximum word rate of 1,536,000 accesses per second. Since each access consumes one (1) OMNIBUS cycle, this then consumes about one-tenth the available data rate for the OMNIBUS interface. Furthermore, it is imperative to retrieve the data promptly so that the internal FIFO is not overrun and the data points become lost. The FIFO helps to mitigate these problems on a transient basis, although care should be used so that software has enough time to retrieve the data under all operating conditions. For these reasons, it is *strongly recommended* that DMA channels be used on DSP cards to service the module data interrupts, since they have the lowest latency and best determinicity. This also relieves the DSP of the onerous task of servicing this high data rate. Chico users need not be concerned, as the data rate is well within the capability of the data-streaming engine.

Analog Input Cabling

The inputs to the AD16 module are differential. Therefore, it is suggested that signals being measured be transmitted to the module deferentially to minimize cable cross-talk and common-mode noise sources. Shielded, twisted-pair cables will give the best results.

Power Consumption

The AD16 module consumes approximately 3.0 watts typically and about 4.5 watts maximum. Forced air-cooling may be necessary under some operating conditions. The maximum rated operating temperature is 55 degrees Celsius ambient, with free air circulation.

CHAPTER 6

AD40 Module

Module Introduction

The AD40 module gives the target processor card two independent channels of ultra-high speed 40 MHz, 12-bit resolution analog input, well suitable for use in high-speed data acquisition, glitch capture, data processing, and control systems. Each channel on the module uses an Analog Devices AD9224 monolithic single supply analog to digital converter with an on chip high performance sample and hold amplifier and voltage reference.

The A/D's use a multi-stage differential pipelined architecture with output error correction logic to provide 12-bit accuracy with no missing or gapping code. A single clock input is used to control all the internal conversion cycles. The digital output data is presented in straight binary output format. On-board circuitry adds gain/ offset error adjustments for each channel to insure accurate measurements.

Each of the A/D channels is equipped with a 1K sample FIFO (64K optional) to allow efficient data collection and transport to the host DSP card. This allows the DSP to collect the data form the A/D as either single points to minimize latency or as a data set of up to the full FIFO size to minimize the interrupt rate to the host DSP. Additional onboard logic supports pre-triggering in which the FIFO is continuously refilled with fresh conversions until an external gating event is detected, after which a programmable number of samples are acquired. Another powerful feature of the AD40 is the external level gating capability. Data acquisition for each channel may be inhibited until the input level exceeds a programmable voltage threshold, after which acquisition is enabled.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

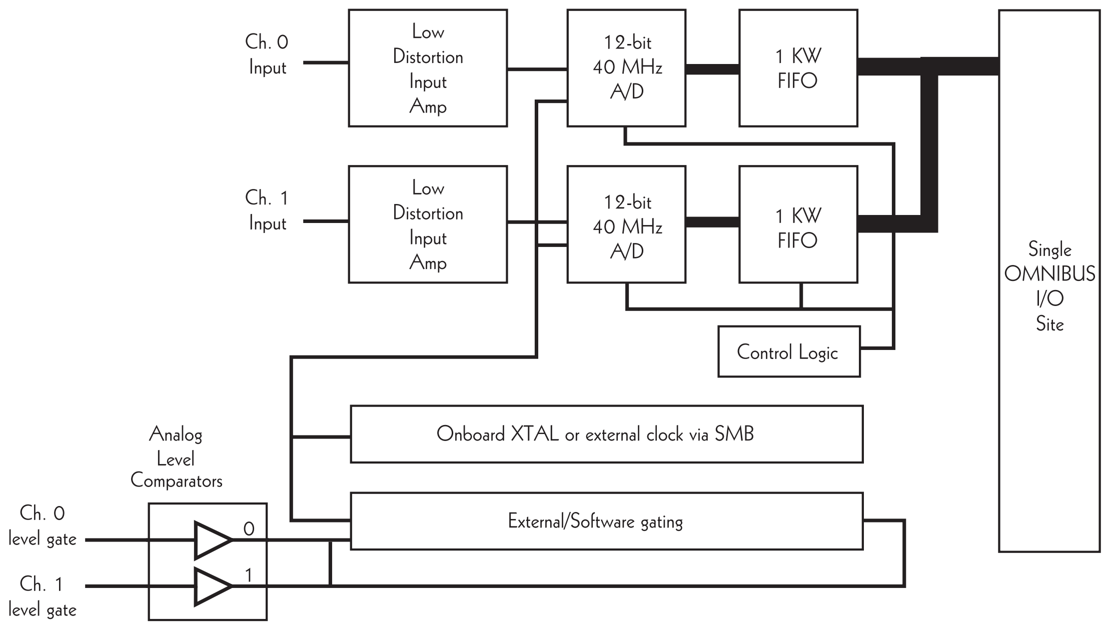


FIGURE 28. **AD40 Block Diagram**

Bus Type:	Compatible with all I.I. products 32-bit. Consumes one interrupt to host DSP. Wait-state depends on host platform	**Sample FIFO's:**	1 K sample standard, memory mapped to DSP. A/Ds are paired on bus as 32-bit numbers - each 16-bit half is an A/D. 64 K optional to support "snapshot" type applications.
Physicals:	OMNIBUS mezzanine card; 2.0" X 4.6"	**Gain Error:**	+/- 1.6 % FSR
A/D Converters:	Two Analog Devices AD9224	**SFDR:**	70 dB
Resolution:	12-bit	**INL:**	+/- 5 LSB
Update Rate:	0 - 40 MHz	**DNL:**	+/- 1 LSB
Pipeline Latency:	Four conversion clocks	**Offset Error:**	Trimmable on each channel - factory calibrated
External Clock Input:	TTL (50 ohm termination optional)	**Aperature Jitter:**	4 ps RMS
External Clock:	0 - 40 MHz	**Aperature Delay:**	1 ns
Analog Input Range:	+/- 1 V	**Input Type:**	Single-ended
SINAD:	fin = 10 MHz = 60 dB	**Input Impedance:**	50 ohms
S/N Ratio:	fin = 10 MHz = 63 dB	**Analog Filter Characteristics:**	None
THD:	fin = 10 MHz = -62 dB	**Conversion Trigger Sources:**	Onboard clock osc. or external clock via SMB into 50 ohm load
Dynamic Range:	fin = 10 MHz = -75 dB	**Pre-Trigger:**	Special logic supports pre-trigger up to the entire FIFO depth.

The target Peripheral Library provides support for each of the AD40 functions, and the following sections give descriptions of the support routines. Refer to the **ad40adc.h** or **ad40int.h** file located in the `c:/<target board>/Include/Target/Analog` director and the **Omnibus.hlp or Zuma.hlp** Windows help file for complete information on each function.

Function Name	Operation
timebase()	Sets sample rate of A/Ds using DDS.
AD40_initialize()	Resets the AD40.
AD40_read_adc_pair()	Reads A/D converter sample results from two channels.
AD40_correct_adc()	Corrects adc value to have the proper range.
AD40_correct_adc_pair()	Corrects adc pair value to have the proper range.
AD40_bleed_fifo()	Dump FIFO contents into memory buffer.
AD40_gate()	Gates acquisitions to all A/D channels.
AD40_trigger_adc_pair()	Configure clock source for A/D channels.
AD40_reset_fifo()	Resets the AD40 FIFOs.
AD40_set_fifo_interrupt_level()	Set FIFO trip point.
AD40_read_status()	Read back of FIFO interrupt flags.
AD40_write_idrom()	Write identification information.
AD40_read_idrom()	Read identification information.
AD40_write_dac()	Set the output of the D/A converter.
AD40_write_analog_trigger()	Sets post-trigger threshold voltage.
AD40_set_pretrigger_length()	Sets pre-trigger buffer length.
AD40_set_trigger_source()	Sets trigger source to external or comparator.
AD40_set_trigger_control()	Sets trigger operating mode.
AD40_normal_omnibus()	Programs module for normal FIFO accesses (For M6x only).
AD40_fast_omnibus()	Programs module for fast FIFO accesses (For M6x only).

TABLE 36. **C Language AD40 Functions**

The AD40 library functions can be divided into several groups:

1. Sample rate control.
2. A/D control.
3. FIFO control.
4. Identification readback.

Sample Rate Control

The sample rate to the AD40 may be precisely controlled via the module site's onboard 9850 DDS timer. The **timebase()** function controls the sampling rate used by all the A/D converters within the application. The DDS clock must be set up to run at one times the desired sample rate due to the nature of the A/Ds.

Alternately, the A/Ds may be clocked using an onboard oscillator or via an external TTL clock. Use the **AD40_trigger_adc_pair()** function to specify the conversion clock source.

A/D Control

The A/Ds on the AD40 begin sampling data after the DDS clock begins running. The FIFO functions control the gating of data.

A/D FIFO Control

To begin storing sampled data in the FIFOs, the **AD40_gate()** function must be called. The gate function allows data to begin entering both FIFOs simultaneously for the A/D channel pair. The FIFOs will continue to fill until they are full at which point no more data will be stored until the FIFOs are reset or emptied.

The FIFOs can be set up to interrupt or flag the processor at three points; full, half full or not empty using the **AD40_set_fifo_interrupt_level()** function and the state of these FIFO flags can be read at any time with the **AD40_read_status()** function.

To read the converted data from the FIFOs, use the **AD40_read_adc_pair()** function. The data comes in stacked on the 32-bit bus with the lower 16-bits being the first channel of the pair and the high 16-bits being the second channel of the pair.

Alternately, the **AD40_bleed_fifo()** function may be used to perform a very fast dump of the FIFO contents into a buffer. This function is much faster than using the **AD40_read_adc_pair()** function in a loop, because it uses a DMA channel to perform the move. However, the data read from the FIFO by this function is not "corrected"; The raw 12-bit A/D samples are placed into the designated buffer and you must use the **AD40_convert_adc_pair()** function to convert the data into signed, twos-complement data for subsequent processing within your application.

The FIFOs are cleared with the **AD40_reset()** function.

Pre-Triggering

The AD40 supports a flexible pre-triggering mechanism on the channel 0 input. When enabled., via the **AD40_set_trigger_control()** function, pre-triggering begins and the channel 0 input voltage is constantly compared to a programmable comparator voltage. When the trigger occurs, a user-specified number of samples are saved in the input FIFO as the pre-trigger information (the size of which is set by the **AD40_set_pretrigger_length()** function), after which new samples are retained until the FIFO fills.

The **AD40_set_trigger_control()** function allows the AD40 to be put into one of five modes. These are listed below:

1. Continuous mode: Data is always taken.
2. Falling Edge: Data is discarded until the signal crosses the threshold in a downward direction. From then data is taken no matter what its level.
3. Rising Edge: Data is discarded until the signal crosses the threshold in an upward direction. Data is taken from then on.
4. Capture Below: Data is taken if it is below the threshold, and discarded otherwise. All data will be below the threshold.
5. Capture Above: Data is taken if above the threshold, and all points below the threshold is discarded.

AD40_set_trigger_source() sets the source of the trigger signal. One option is to use the comparator as described above. The other option is to use an external digital gate signal from the I/O connector to give the edge and/or levels that the hardware requires.

AD40_write_analog_trigger() sets the threshold voltage used by the comparator.

Identification Readback

The **AD40_read_idrom()** function is used to read the identification ROM on the AD40 to check its identity and revision level. The function fills out an `AD40_ID` structure with the information stored in the ID ROM on the module. The `AD40_ID` structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string “AD40”).
2. Module revision level (single character revision level).
3. Checksum.

This data is preprogrammed at the factory and should not be altered by the user.

Memory Mapping

The AD40 module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** or the **ad40.h** file included with the Zuma Tools DSP software library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	Read/ Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
A/D Data Readback	R	IOMOD0 + 0	IOMOD4 + 0	IOMOD8 + 0
ADC Clock Select	W	IOMOD0 + 4	IOMOD4 + 4	IOMOD8 + 4
Pre-trigger Count	W	IOMOD0 + 5	IOMOD4 + 5	IOMOD8 + 5
Gate Source	W	IOMOD0 + 7	IOMOD4 + 7	IOMOD8 + 7
FIFO Flag Readback	R	IOMOD1 + 3	IOMOD5 + 3	IOMOD9 + 3
FIFO Flag Select	W	IOMOD1 + 3	IOMOD5 + 3	IOMOD9 + 3
FIFO Reset	W	IOMOD1 + 4	IOMOD5 + 4	IOMOD9 + 4
Analog Trigger Level D/A Control	W	IOMOD1 + 6	IOMOD5 + 6	IOMOD9 + 6
IDROM SDA	R/W	IOMOD3 + 0	IOMOD7 + 0	IOMOD11 + 0
IDROM SCK	R/W	IOMOD3 + 1	IOMOD7 + 1	IOMOD11 + 1

TABLE 37. **AD40 Memory Map**

Interrupt Usage

The AD40 uses a single OMNIBUS interrupt input to the baseboard processor to notify the CPU of a programmed FIFO level status. This interrupt pulse may be used to trigger either CPU interrupts or DMA synchronization events, where applicable, in order to move digital data from the AD40 to the host processor's memory.

The AD40's interrupt mode selection is programmable using the FIFO Flag Select register shown below. The module may be programmed to assert interrupts on FIFO not empty, FIFO half-full, or FIFO full conditions. At device power up or after reset the module initializes with interrupt drive turned off, to avoid potential hardware conflicts with other modules or external hardware. The table below shows the control register bit definition. The states of the FIFO flags can be read back by use of the FIFO Flag Readback Register.

Bit	31-3	2	1	0
	Reserved	FULL	HALF_FULL	NOT_EMPTY

FIGURE 29. **AD40 FIFO Flag Select Register**

Bit Field Name	Function
NOT_EMPTY	1 = interrupt on FIFO not empty
HALF_FULL	1 = interrupt on FIFO half full
FULL	1 = interrupt on FIFO full

TABLE 38. **AD40 FIFO Flag Select Register Definition**

Note that only one interrupt mode should be used at one time (i.e. only one bit should be set high in the FIFO Level Interrupt Mode register).

Pin Connector I/O

The AD40 output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* for additional details on how to connect to the AD40's I/O pins.

AD40 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50-Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1..45	29..50, 78..100	4..25, 53..75	1..25, 31..50	1..45	1..5, 9..13	Reserved
46	28	3	30	46	6	External Gate Input
47	77	52	29	47	14	Analog Ground
48	27	2	28	48	7	Analog Input Channel 1 (Optional, see below)
49	76	51	27	49	15	Analog Ground
50	26	1	26	50	8	Analog Input Channel 0 (Optional, see below)

TABLE 39. **AD40 I/O Connector Pinout**

The AD40 provides two means to connect analog signals to the signal inputs: via the I/O connector pinouts given above or through MCX style coaxial connectors. The I/O connector allows for easy external interfacing through the existing OMNIBUS host interface, while the coaxial connectors mate easily with existing coax hookups.

On-board resistor steering allows the user to select which of the two possible input methods to use. By default, the AD40 comes configured to accept input signals on the coax connectors only. Connector (J2) is the input for A/D channel 0, while connector (J3) is the input to channel 1. By changing the following zero ohm jumper positions the module may be configured to receive inputs on the I/O connector pins:

A/D Channel	Desired Input type	Required Jumper Setup
0	Coaxial (via J2)	R11 installed, R14 removed
	I/O Connector (pin 50)	R14 installed, R11 removed
1	Coaxial (via J3)	R27 installed, R30 removed
	I/O Connector (pin 48)	R30 installed, R27 removed.

The coaxial connectors are right angle MCX type (Johnson part number 133-3701-311). Innovative recommends the use of AMP straight through coax plugs (part number 829550-2) for connections to these inputs. The coax shell carries the analog ground reference, to which the center conductor signal should be referenced.

The AD40 also provides a third coaxial connector (J1), which accepts an external sample clock for use in controlling the conversion rate of the A/D converters (see below for details).

Functions

A/D Converters

Analog Devices AD9224 A/D converters are used on the AD40 to implement two channels of 12-bit conversion at sampling rates up to 40 MHz. The device returns 12-bit straight binary codes (full scale range of 0 to 4095) at the selected conversion rate. No clock division is used to implement the internal digital filters within the device, so resultant data samples are delivered at the full conversion rate.

Since the AD9224 is a multistage pipelined device, digital data delivered at the outputs lags the input signal by the device's pipeline delay (3 conversion clock cycles). In the FIFO-buffered single acquisition applications for which the AD40 is intended, this delay is not ordinarily an issue.

Input Circuitry

The AD40 uses a high speed single-ended differential conversion design to drive the single-ended input signal into the AD9224 converter's differential inputs. Each A/D converter's input circuitry converts the single-ended +-1 volt input to a unipolar differential signal set suitable for digitalization by the converter. The input circuitry is noninverting, resulting in a 0 to 4095 count conversion range for the –1V to +1V analog input range.

Two poles of rolloff are provided on the inputs for high frequency noise rejection. One pole sits at slightly over 100 MHz, while the other is slightly above the maximum Nyquist frequency, at 21 MHz.

The following figures give the schematics for the input circuitry for each A/D channel.

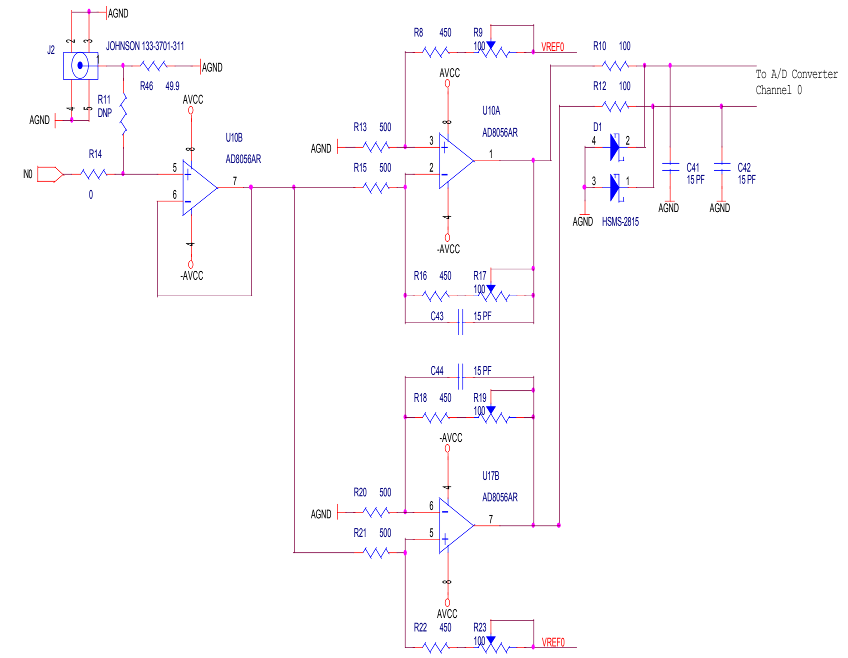


FIGURE 30. AD40 A/D Channel 0 Input Circuitry

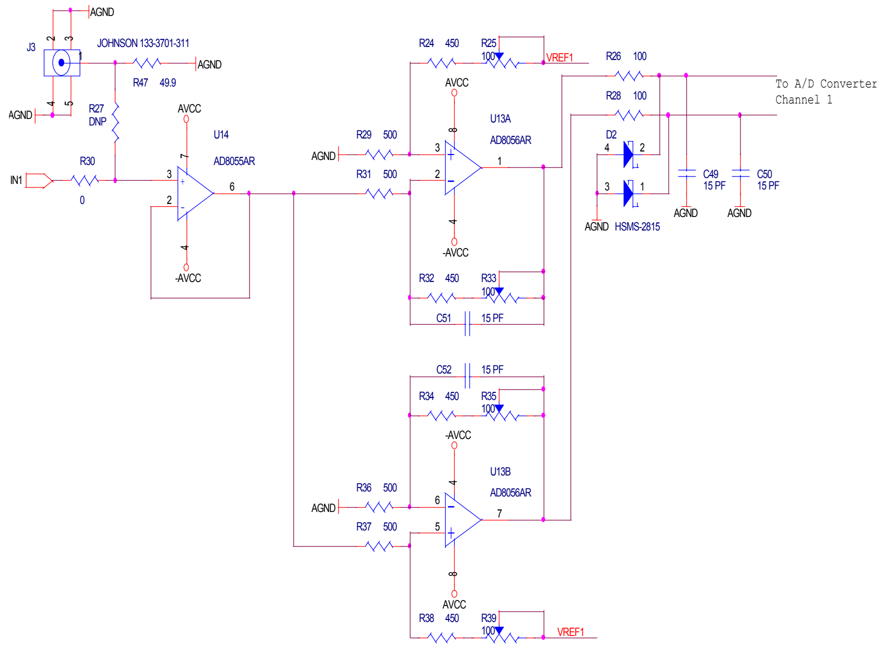


FIGURE 31. AD40 A/D Channel 1 Input Circuitry

The input stages are trimmed at the factory for full scale digital output given a +-1V no offset input. The trim procedure is as follows:

1. Connect a function generator (HP 33120A, SRS DS360, or equivalent) to each AD40 input via a BNC to MCX cable. Set the generator for 2Vp-p 10 kHz sine, 0V offset.

2. Using an HP 34401A meter set to AC V mode, measure the input voltage level at the input connector (it should be close to 0.707 VRMS). Measure the following points with the meter and use the corresponding trim pots to adjust the AC voltage to within +-0.001 VRMS of the input voltage. A silkscreen print of the AD40 is included at the bottom of the test sequence to help in locating the ICs and trim pots.

Channel	Measure point	Adjust
0	U10 pin 1	R17
	U17 pin 7	R19
1	U13 pin 1	R33
	U13 pin 7	R35

   Use the input jack's ground pins as a ground reference for the measurement. This is easier if a small (1/2") 20 or 22 gauge wire is temporarily soldered to one of the ground pins during testing.

3. Set the function generator amplitude to 0V and the meter to DC V mode. Measure the following points and adjust each to 2.000V +-0.001V.

Channel	Measure point	Adjust
0	U10 pin 1	R9
	U17 pin 7	R23
1	U13 pin 1	R25
	U13 pin 7	R39

The following diagram gives the silkscreen print for the AD40, to aid in identifying the component reference designators.

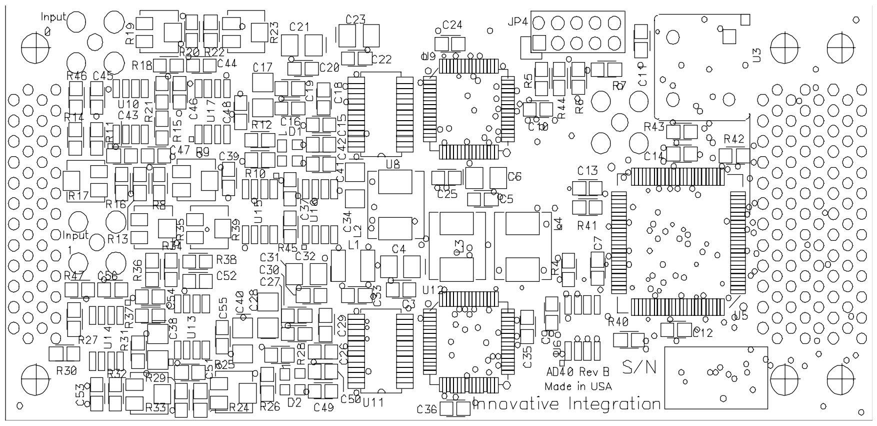


FIGURE 32. AD40 Silkscreen

Conversion Clock Sources

Conversion clock sources are controlled through software via the A/D Conversion Clock Select register. The register allows software to select from three different timebase sources: the OMNIBUS DDS synthesizer clock, an external clock provided via the (J1) connector, or an optional onboard TTL oscillator (provided by the user). The register bit definition is given below:

Bit Number:	31-3	2	1	0
Bit Field:	Reserved	XTAL	EXTERNAL	DDS

FIGURE 33. AD40 A/D Conversion Clock Select Register

Bit Field Name	Function
DDS	1 = selects OMNIBUS DDS clock source
EXTERNAL	1 = selects external clock source via connector J1
XTAL	1 = selects optional onboard TTL oscillator

TABLE 40. AD40 A/D Conversion Clock Select Register Definition

Setting the corresponding bit in the Clock Select register connects the source to both A/D converters as their conversion clock. The converters are triggered simultaneously from the same clock source: independent conversion of each A/D is not supported by the AD40.

If the DDS timebase source is used, then the OMNIBUS host's timer must be programmed for the appropriate sample frequency. Use of the DDS timebase as a conversion source may limit the maximum conversion rate frequencies. For example, on 9850 equipped DSP hosts the DDS maximum frequency is 25 MHz, limiting the AD40's maximum conversion rate to the same frequency when used in DDS clock source mode with those hosts.

If the external clock source is used, a clean TTL-compatible clock signal must be connected to connector (J1) via 50 ohm coaxial cable (typically AMP 829550-2 connectors terminating RG178 cable). The input connector is terminated to digital ground with a 49.9 ohm resistor (R43). Please note that for the best results, the *digital* ground node present on connector (J1) should not be connected in external equipment, the *analog* ground node present on the I/O connector and on the analog coax inputs (J2) and (J3).

If the onboard oscillator clock source is selected, the user must attach a 5V powered TTL compatible half-size oscillator at location U3. The AD40 is compatible with oscillators from CTX (MXO-45 series), Epson (SG-531 series), and Ecliptek (EC1100 series). A series resistor is provided in line with the oscillator output (R41, nominally 0 ohms) for use in terminating the onboard clock source.

The AD9224 converters are sensitive to clock jitter, so the quality of the external or oscillator clocks is extremely important, especially in very high speed applications (approaching 40 MHz sample rates). The input clock must observe the AD9224 timing specifications (12.37 ns minimum high and low time). Refer to the AD9224 data sheet for additional details.

Please note that since the AD9224 converter is a pipelined architecture it is sensitive to the clock starting and stopping during acquisitions. To avoid disturbing the acquired data, make sure that the clock source is always running at the same frequency across the complete acquisition sequence.

Software and Hardware Gating

Input signal acquisition is enabled via the AD40's FIFO software gate register. Writing a one (1) to the register enables acquisition of converted analog data by the FIFOs. The register is reset automatically to 0 on power up or OMNIBUS reset, and must be written with a one (1) before data will be acquired by the FIFOs.

External hardware gating is also supported by the AD40 via the external gate input on the I/O connector. If pulled low by external hardware, this signal will also enable acquisition of A/D data by the FIFOs. There are no timing requirements for the external gate input: it is internally re-timed to the A/D acquisition clock. The external gate input us pulled up to digital 5V via a 10K resistor.

Analog Triggering

The AD40's analog trigger feature allows the module to control acquisition of incoming signals by triggering on the state of the signal present on the channel 0 input. Software programmable trigger level and slope controls allow the AD40 to "search" for a particular edge type and voltage level in the same way as oscilloscopes and other test instruments trigger off incoming signals. Flexible FIFO pre- and post-trigger controls allow software to set the length of time captured before and after the triggering event. These features make the AD40 an excellent basis for computer controlled oscilloscope and test instrumentation.

The following figure gives a block diagram of the analog trigger functionality. The positive input signal from A/D channel 0 is buffered and driven, along with a programmable D/A-generated voltage reference, into a comparator which detects when the input signal passes the D/A reference point. This information is converted into a dual slope digital signal by the comparator, which is used by onboard logic to control the acquisition of data into the FIFOs.

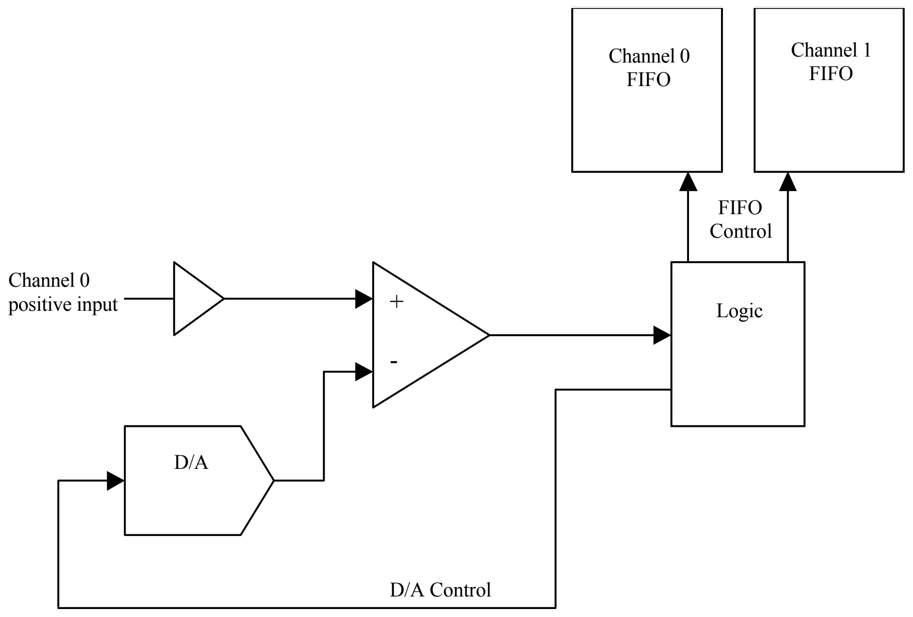


FIGURE 34. AD40 Analog Trigger Block Diagram

When placed into analog trigger mode, the AD40's onboard logic waits for the programmed trigger event to occur while simultaneously controlling the FIFOs to maintain a pre-trigger memory buffer. Once the trigger event occurs, the FIFOs are allowed to continue acquiring digitized A/D data until they fill. The result is a FIFO-length memory buffer with a portion of the recorded data occurring before the trigger event and a portion recorded from immediately after the event.

The pre-trigger count, analog trigger level, and trigger edge type and trigger arming are all controlled through AD40 registers. The Pre-trigger Count register is used by software to control the amount of samples reserved for pre-trigger information.

The analog trigger level is set by the D/A reference which is in turn controlled via the Analog Trigger Level D/A Control register. This register provides for direct software control of the serial interface to the Linear Technology LTC1451 D/A converter used to implement the reference. The following figure gives the definition for the register.

Bit Number:	31-3	2	1	0
Bit Field:	Reserved	CLOCK	LOAD	DATA

FIGURE 35. AD40 Analog Trigger D/A Control Register

Bit Field Name	Function
DATA	D/A data control
LOAD	D/A load control
CLOCK	D/A clock control

TABLE 41. AD40 Analog Trigger D/A Control Register Definition

Bit values written to the register are sent directly to the D/A converter, allowing software to control the D/A output by using bit sequences compatible with the LTC1451's input interface. The LTC1451 is a 12-bit device with a 0 to 4.095 volt output range accepting straight binary input data in the range 0 to 4095. Since the A/D front end interface converts the +-1V input signal to a differential noninverting DC offset swing of +1 to +3 volts for compatibility with the A/D converter's input requirements. The D/A output reference for use in analog triggering should also have a +1 to +3 volt swing. This means that the full D/A dynamic range is not utilized and that the D/A converter never has an input code outside the range of approximately 1000 to 3000 or (about half its dynamic range, resulting in an effective 11-bit reference resolution). The following table gives D/A data values for a range of trigger voltage settings and shows the relationship between input signal levels and D/A voltage levels.

Desired Input Trigger Voltage	Effective A/D Input Voltage	Required D/A Reference Voltage	Required D/A Data Value
+1.0	+3.0	+3.0	3000
+0.5	+2.5	+2.5	2500
0	+2.0	+2.0	2000
-0.5	+1.5	+1.5	1500
-1.0	+1.0	+1.0	1000

The trigger and gating functions of the analog trigger are controlled through the Gate Source Control register. This register allows software to set how the AD40 searches for a trigger to initiate data taking and how data is acquired after sampling begins.

Bit	31-5	4	3	2 1	0
	Reserved	Type	External	Gate	Armed

FIGURE 36. AD40 Gate Source Control Register

Bit Field Name	Function
Armed	Write of 1 arms the analog trigger
Gate	bits = 00: Edge Trigger on negative-going slope
	bits = 01: Edge Trigger on positive-going slope
	bits = 10: Capture voltages *below* threshold
	bits = 11: Capture voltages *above* threshold
External	0 = Comparator Trigger, 1 = External Digital Trigger
Type	0 = Event, 1= Continuous

TABLE 42. AD40 Gate Source Control Register Definition

The Gate field of the Gate Source Control register allows software to select the triggering and gating mode the AD40 will use. There are four options, which are divide into 2 types of acquisitions. The first two options are edge triggers. In this mode, the AD40 module works like an oscilloscope trigger: The signal is scanned for a moment when the signal crosses the programmed threshold and then from that point forward the signal is loaded into the FIFO. The direction of the threshold crossing must be selected to be a positive-going crossing or a negative-going crossing. If pre-triggering is in use, the pre-trigger samples are the N samples before this crossing.

The other mode gates the signal at all times after triggering. The AD40 only acquires samples where the data is above or below a set threshold voltage. For example, when the bits are 11, all samples above the threshold will be acquired and all those below are skipped. A sine wave input would appear to produce a connected series of humps, with all values at the threshold discarded. Note that if the signal is above the threshold at the start, it must pass below the threshold and then cross above it before data is collected. The below threshold situation is similar, except the data accepted is all below the threshold and all data above the threshold is discarded. If a pre-triggering mode is in use, the N sample before the starting point are collected *without* gating any of the points. Therefore in the pre-trigger region, points on the ‘wrong side’ of the threshold can be collected.

The External field selects the source of the signal used as the trigger for the AD40 logic. If the bit is 0, the channel 0 signal is compared to the DAC threshold level to produce the trigger for the AD40. If the signal is above the threshold, for instance, the comparitor produces a 1. This comparator signal is then processed for leading edge, trailing edge and level characteristics as defined by the Gate field.

When the External field is set to 1, the comparator threshold level is ignored and an external digital input signal is used instead. When this signal is high, you are above threshold, and when low you are below threshold. The signal used is the External Gate Input (see Table 39 on page 107).

If the Type field is set to 1, all gating features are ignored.

A 1 bit is written to the Armed bit to arm the analog trigger and allow it to begin searching the analog input signal for the programmed trigger event (voltage plus slope). When programming the register, it should always be written twice (first with the edge type, then with the edge type plus the ARM bit set) in order to ensure that the correct slope is detected.

FIFOs

1K (64K optional) sample depth FIFOs are provided for capturing analog information at high rates directly from the A/D converters, and to act as sample buffer memory when using the analog trigger feature. The FIFO buffers may be reset under software control, and their depth flags can be read back by software as well as used as interrupt sources to the OMNIBUS host card.

The FIFO reset is controlled by the FIFO Reset register. A write of any data value is sufficient to send a reset pulse to both FIFOs. Resetting the FIFOs clears their internal depth counters and resets the flags to indicate an empty state.

The FIFO flag values may be read via the FIFO Level Readback register. The following diagram shows the register definition.

Bit Number:	31-3	2	1	0
Bit Field:	Reserved	FULL	HALF_FULL	NOT_EMPTY

FIGURE 37. AD40 FIFO Level Readback Register

Bit Field Name	Function
NOT_EMPTY	0 = empty, 1 = not empty
HALF_FULL	0 = less than half full, 1 = half full
FULL	0 = not full, 1 = full

TABLE 43. AD40 FIFO Level Readback Register Definition

In addition to polling the level status, the OMNIBUS host may elect to use one of the FIFO status bits as an interrupt source. See the Interrupt section for more information.

A 64K sample depth option is available for applications requiring deeper sample memories. Contact Innovative Integration for details.

Initialization Issues

Normal Acquisition Mode

The following initialization order should be used when the AD40 is being set up for non-analog trigger based data acquisition.

1. Turn the A/D gate off.

2. Reset the FIFOs.
3. Set the A/D clock source.
4. Setup OMNIBUS host interrupts and set the AD40 interrupt mode (if used).
5. Initialize the clock source.
6. Turn the A/D gate on.

Analog Trigger Acquisition Mode

The following initialization order should be used when the AD40 is being set up for analog trigger based data acquisition.

1. Turn the A/D gate off.
2. Reset the FIFOs.
3. Setup OMNIBUS host interrupts and set the AD40 interrupt mode (if used).
4. Set the A/D clock source.
5. Select the Gate Source but do **not** arm.
6. Set the pre-trigger count (the number of points in pre-trigger +1).
7. Set the analog trigger level voltage to the D/A.
8. Arm the analog trigger (remember to preserve all the other bits!).
9. Turn the A/D gate on.

AD40 Notes

Some other items to keep in mind:

1. The AD40 will not give an interrupt flag until the pre-trigger count is reached.
2. For continuous mode, the pre-trigger count must be 0.
3. You must disarm between runs and reload the pre-trigger counter. These values in these registers are not saved.
4. The FIFO flags can be read back from the Flag Readback Register.

CHAPTER 7

AIX (AIX20) Module

Module Introduction

The AIX (AIX20) OMNIBUS module provides the target card processor with four channels of very high speed 2.5 MHz (20 MHz), 16 bit (12 bit) resolution analog input to digital output conversion. This makes it well suited for high-speed data acquisition, glitch capture, data processing, and control systems. The AIX (AIX20) module uses two pairs of Analog Devices AD9260 A/D's to provide an excellent dynamic range over a wide bandwidth. The A/D uses unique pipeline architecture to sample the data input at up to 20 MHz (20MHz) and digitally filters averages the data output for accurate results. The A/D delivers 2.5 MHz (20 MHz) data from the filtered pipeline that is only eight samples deep, which results in low data latency. Additionally, each A/D channel has an anti-alias filter and gain/offset error adjustment to maintain accurate data measurements.

An eight (8) kword FIFO on each channel separates the A/D's from the data bus allowing the DSP time for other tasks while the FIFO's are filling or emptying. The FIFOs allow the DSP to collect the data from the A/D's as single points or as a data set of up to 8K sample size. This reduces the interrupt rate to the host DSP allowing for highly efficient data collection. In addition, full speed, continuous data acquisition is supported to the DSP.

The AIX20 is identical to the AIX Module except for the A/D is connected in the 1x decimation mode turning it into a 12 bit 20 MHz A/D with no anti-aliasing filters. All other features are identical to the AIX module and for the purpose of this text, the "20" has been dropped when referring to functional codes (**AIX20_gate()** will be referred as **AIX_gate()**).

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

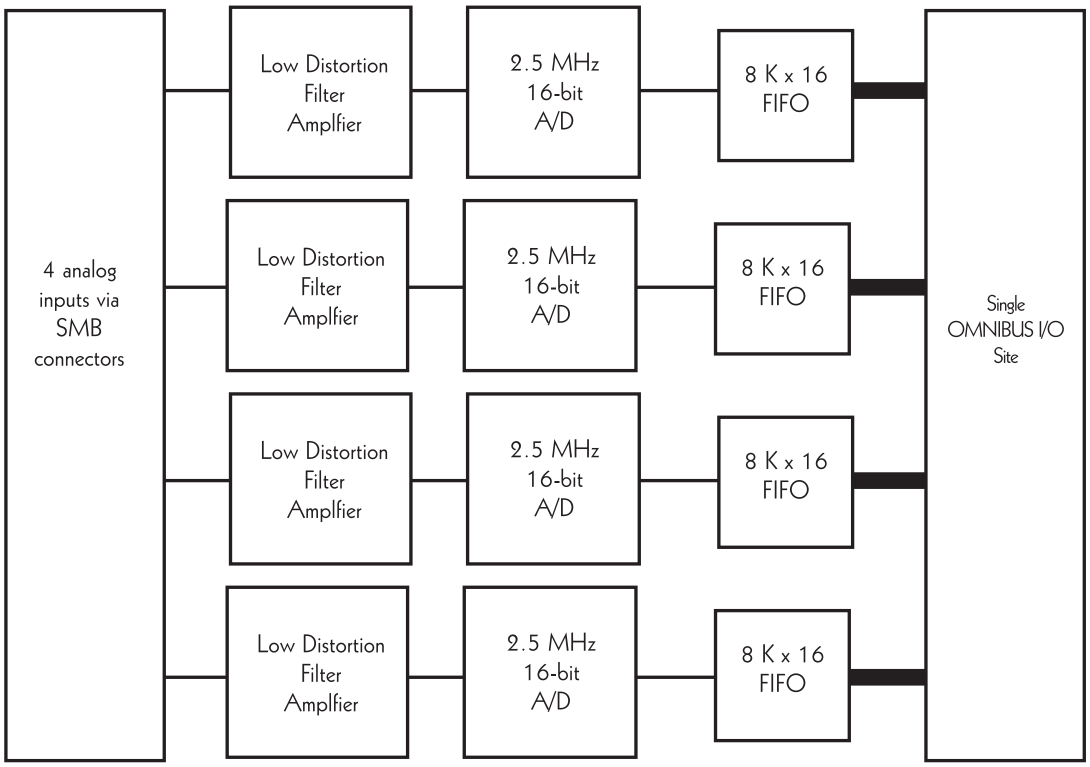

4 analog inputs via SMB connectors
Low Distortion Filter Amplfier
2.5 MHz 16-bit A/D
8 K x 16 FIFO
Low Distortion Filter Amplfier
2.5 MHz 16-bit A/D
8 K x 16 FIFO
Low Distortion Filter Amplfier
2.5 MHz 16-bit A/D
8 K x 16 FIFO
Low Distortion Filter Amplfier
2.5 MHz 16-bit A/D
8 K x 16 FIFO
Single OMNIBUS I/O Site

FIGURE 38. AIX Block Diagram

Bus Type:	Compatible with all I.I. products; 32-bit. Consumes one interrupt to host DSP. Wait-states depend on host platform.	**Input Type:**	Single-ended
Physicals:	OMNIBUS mezzanine card 2.0" X 4.6"	**Input Impedance:**	50 ohms | | 390 pF
A/D Converters: (four A/D chips)	Analog Devices AD9260 Pipelined architecture (8 samples) for low data latency. Each converter channel has independent filtering and error trims.	**Digital Filter Characteristics**	Passband Ripple 0.004 dB; Passband Attenuation; 85.5 dB; Passband 1.01 x (fclock/20 MHz) MHz; -3 dB transition; 1.2 x (fclock/20 MHz) MHz; Stopband 1.49 x (fclock/20 MHz) MHz min.; 18.51 x (fclock/20 MHz) MHz max.
Resolution:	16-bit	**Analog Filter Characteristics:**	Single pole filter, -3 dB set at 30 MHz
Update Rate:	2.5 MHz	**Conversion Trigger Sources:**	Timers
External Clock	Eight times the sample rate.	**Aperature Jitter:**	2 ps
Settling Time	15 us max. to 0.0007% (no external filtering)	**Sample FIFO's**	8 K samples standard; Memory-mapped to DSP; A/Ds are paired on bus as 32-bit numbers - each 16-bit half is one A/D
Analog Input Range:	+/- 2V, custom ranges may be special ordered.	**Gain Error:**	Trimmable on each channel - factory calibrated
S/N Ratio:	85 dB @ 2.5 MHz	**Diff. Linearity Error:**	0.5 LSB typical
THD:	90 dB	**Offset Error:**	Trimmable on each channel - factory calibrated
Dynamic Range:	95 dB		

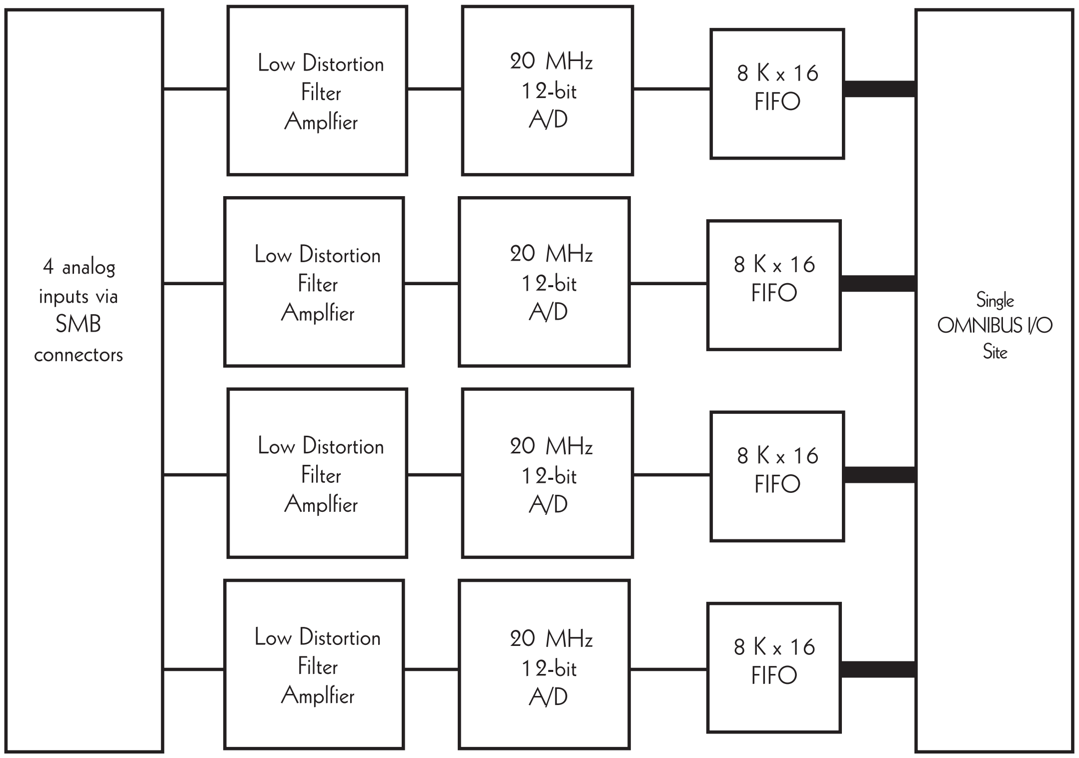

Low Distortion Filter Amplfier
20 MHz 12-bit A/D
8 K x 16 FIFO
Low Distortion Filter Amplfier
20 MHz 12-bit A/D
8 K x 16 FIFO
Low Distortion Filter Amplfier
20 MHz 12-bit A/D
8 K x 16 FIFO
Low Distortion Filter Amplfier
20 MHz 12-bit A/D
8 K x 16 FIFO
4 analog inputs via SMB connectors
Single OMNIBUS I/O Site

FIGURE 39. **AIX20 Block Diagram**

Bus Type:	Compatible with all I.I. products; 32-bit. Consumes one interrupt to host DSP. Wait-states depend on host platform.	**Input Type:**	Single-ended
Physicals:	OMNIBUS mezzanine card 2.0" X 4.6"	**Input Impedance:**	50 ohms
A/D Converters: (four A/D chips)	Analog Devices AD9260 Pipelined architecture (8 samples) for low data latency. Each converter channel has independent filtering and error trims.	**Sample FIFO's**	8 K samples standard; Memory-mapped to DSP; A/Ds are paired on bus as 32-bit numbers - each 16-bit half is one A/D
Resolution:	12-bit (+/- 1848 counts)	**Analog Filter Characteristics:**	Single pole filter, -3 dB set at 12 MHz
Update Rate:	20 MHz	**Conversion Trigger Sources:**	Timers
External Clock	Up to 20 MHz	**Aperature Jitter:**	2 ps
Settling Time	15 us max. to 0.0007% (no external filtering)	**Dynamic Range:**	85 dB
Analog Input Range:	+/- 2V, custom ranges may be special ordered.	**Gain Error:**	Trimmable on each channel - factory calibrated
S/N Ratio:	60 dB	**Diff. Linearity Error:**	0.25 LSB typical
THD:	75 dB	**Offset Error:**	Trimmable on each channel - factory calibrated

The target Peripheral Library provides support for each of the AIX & AIX20 functions, and the following sections give descriptions of the support routines. Refer to the **aixadc.h** or **aixint.h** files located in the `c:/<target board>/Include/Target/Analog` director and the **Omnibus.hlp or Zuma.hlp** Windows help file for complete information on each function.

AIX Function Name	**AIX20 Function Name**	**Operation**
timebase()	timebase()	Sets sample rate of A/Ds using DDS.
AIX_read_adc_pair()	AIX20_read_adc_pair()	Reads A/D converter sample results from two channels.
AIX_enable_fifo()	AIX20_enable_fifo()	Gates acquisitions from selected A/D pair into FIFO.
AIX_gate()	AIX20_gate()	Gates acquisitions to all A/D channels.
AIX_reset_fifo()	AIX20_reset_fifo()	Resets the AIX FIFOs.
AIX_bleed_fifo()	AIX20_bleed_fifo()	Bleeds data from FIFO into ram buffer.
AIX_set_fifo_interrupt_level()	AIX20_set_fifo_interrupt_level()	Set FIFO trip point.
AIX_read_status()	AIX20_read_status()	Read back of FIFO interrupt flags.
AIX_write_idrom()	AIX20_write_idrom()	Write identification information.
AIX_read_idrom()	AIX20_read_idrom()	Read identification information.
AIX_initialize()	AIX20_initialize()	Resets the AIX module.
AIX_fast_omnibus()	AIX20_fast_omnibus()	Programs module for fast FIFO acesses (For M6x only).
AIX_normal_omnibus()	AIX20_normal_omnibus()	Programs module for normal FIFO acesses (For M6x only).
	AIX20_trigger_adc()	Selects a hardware trigger source for A/D conversions.

TABLE 44. C Language AIX and AIX20 Functions

The AIX & AIX20 library functions can be divided into the following groups:

1. Sample rate control.
2. A/D control.
3. FIFO control.
4. Identification readback.

Sample Rate Control

AIX

The sample rate to the AIX may be precisely controlled via the module site's on board 9850 DDS timer. The **timebase()** function controls the sampling rate used by all the A/D converters within the application. The DDS clock must be set up to run at eight times the desired sample rate due to the oversampling of the A/Ds.

AIX20

Similarly, the AIX20 is controlled via the module's onboard 9850 DDS timer or by an external clock. When using the onboard DDS clock, the **timebase()** function controls the sampling rate used by all the A/D converters within the application. The DDS clock must be set up to run at the same rate as (one times) the desired sample rate due to no oversampling of the A/Ds when run in the high-speed mode. Note that if the external clocking is used, the sample rate will be equal to the external clock rate. Regardless of the clock source used, the **AIX20_gate()** function controls whether data is clocked from the A/Ds into the FIFOs.

In addition to the onboard DDS, an external clock source can be used. Although external clocking of the A/D converters is permitted with this module, the connection scheme is not designed to receive high frequency clocks properly due to the size of connectors and circuitry that would be necessary. The best way to do external clocking at rates higher than 1 MHz is to use external drivers and receivers with the appropriate terminations and have the receiver board as close to the connection on the OMNIBUS module as possible.

The AIX20 design adds an additional control register, which is used by software to select the source of the converter clock signal. The register may be set to disable the clock, select the baseboard DDS clock source, or select the external clock source provided on the OMNIBUS I/O connector. The following table gives the memory mapping for the clock control register.

Function	Read/ Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address	Value
Clock Selection	W	IOMOD0 + 0x8	IOMOD4 + 0x8	IOMOD8 + 0x8	0 – Clock Disabled 1 - DDS Clock 2 – External Clock

FIGURE 40. **AIX (AIX20) Clock Control Register**

A/D Control

The A/Ds on the AIX begin sampling data after the DDS clock begins to run. The FIFOs control the gating of data. Data will not be clocked from the A/D's into the FIFOs until the **AIX20_gate()** function is called.

AIX20 Only: The A/Ds on the AIX20 begin sampling data after either the DDS clock or external clock begins running.

FIFO Control

To begin storing sampled data in the FIFOs, both the **AIX_fifo_trigger()** & **AIX_gate()** functions must be turned on. The gate function allows data to begin entering all four FIFOs simultaneously on previously triggered pairs. The FIFOs will continue to fill until they are full at which point no more data will be stored until the FIFOs are reset or emptied.

The FIFOs can be set up to interrupt or flag the processor at three points; full, half full, or not empty using the **AIX_set_fifo_interrupt_level()** function. The state of these FIFO flags can be read at any time with the **AIX_read_status()** function. The FIFOs are cleared with the **AIX_reset()** function.

AIX Only: To read the converted data from the FIFOs, use the **AIX_read_adc_pair()** function, which takes the pair number to read as an argument (pair 0 for channels 0 & 1, pair 1 for channels 2 & 3). The data comes in stacked on the 32-bit bus with the lower 16 bits being the first channel of the pair and the higher 16 bits being the second channel of the pair.

AIX20 Only: To read the converted data from the FIFOs, use the **AIX20_read_adc_pair()** function, which takes the pair number to read as an argument (pair 0 for channels 0 & 1, pair 1 for channels 2 & 3). The data comes in stacked on the 32-bit bus with the lower 12 bits being the first channel of the pair and the high 12 bits being the second channel of the pair.

NOTE: That the AIX20 A/D drives its conversion results onto data bits sixteen through four on the data bus, *not* bits eleven through zero. This means that the conversion results read from the module *appear* to have sixteen bits of dynamic range. In reality, only the *upper twelve bits* contain meaningful A/D conversion results. Furthermore, due to architectural restrictions of the A/D running in the fast, 12-bit mode, the AIX20 is incapable of converting voltages which lie within 10% of the rail voltages (+/-2 Vdc) of the analog inputs. Thus, the usable input span of the AIX20 is +/-1.804 volts, over which the A/D produces output codes ranging between +29568 and -29568 counts. For more in depth information on the AD9260 refer to the data sheet, which can be found on Analog Devices website (www.analog.com).

Identification Readback

The **AIX_read_idrom()** function is used to read the identification ROM on the AIX (AIX20) to check its identity and revision level. The function fills out an **AIX_ID** structure with the information stored in the ID ROM on the module. The **AIX_ID** structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string “AIX”).
2. Module revision level (single character revision level).
3. Checksum.

This data is preprogrammed at the factory and should not be altered by the user.

Memory Mapping

The AIX (AIX20) module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** or the **aix.h** file included with the host board’s Development Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	Read/ Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address	Value
A/D Pair 0 Data Read	R	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0	
A/D Pair 0 Enable FIFO Fill	W	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0	1 - Enables, 0 Disables
A/D Pair 1 Data Read	R	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1	
A/D Pair 1 Enable FIFO Fill	W	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1	1 - Enables, 0 Disables
A/D Pairs 0 & 1 Gate Enable	W	IOMOD0 + 0x2	IOMOD4 + 0x2	IOMOD8 + 0x2	1 - Enables, 0 Disables
FIFO Interrupt Flag Selection	W	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3	1 – Not Empty 2 – Half Full 4 - Full
Interrupt Flag Readback	R	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3	0 – No Data Available 1 – FIFO Not Empty 3 – FIFO Half Full 7 – FIFO Full
FIFO Reset	W	IOMOD0 + 0x4	IOMOD4 + 0x4	IOMOD8 + 0x4	Any Value
IDROM SDA	R/W	IOMOD3 + 0x0	IOMOD7 + 0x0	IOMOD11 + 0x0	
IDROM SCK	W	IOMOD3 + 0x1	IOMOD7 + 0x1	IOMOD11 + 0x1	
A/D Clock Select Register (AIX20 only)	W	IOMOD0 + 0x8	IOMOD4 + 0x8	IOMOD8 + 0x8	1 - Selects DDS 2 - Selects External Clock

TABLE 45. **AIX (AIX20) Memory Map**

Interrupt Usage

The AIX (AIX20) uses an external interrupt to the baseboard processor to signal the appropriate FIFO flag. This interrupt may be used to trigger CPU interrupts or to begin a DMA to transfer data from the FIFO to local or global memory.

The types of FIFO interrupt what can be sent to the processor are selectable. Typically, users will want to use the half full interrupt which is the default at power up. This is to allow ample time between the receipt of the flag and the start of the data movement without developing data gaps.

Note to M44 users: Typically the AIX (AIX20) module should be installed in module site 0. This allows the card to function properly with the interrupt jumpers in their default position on rev. D and earlier M44 host cards. This is due to the AIX (AIX20) module using external interrupt 0 for module site 0 and external interrupt 2 for module site 1. To use the AIX (AIX20) in module site 1 on early M44 cards, a jumper wire must be placed between pins 20 and 11 of JP13 on the M44card. Note that these restrictions are not true of later revision M44 cards (which provide a modified interrupt jumper configuration that allows easy selection of external interrupt 2 as a processor interrupt source) and later OMNIBUS host card designs (SBC62, M62, etc.). Since these cards all provide for software programmable interrupt settings, they do not use interrupt jumpers.

Pin Connector I/O

The AIX (AIX20) output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual* for additional details on how to connect to the AIX's (AIX20) I/O pins.

AIX Omnibus M173280-3 Connector Pin Number	**ChicoPlus & Hombre Baseboard 100 Pin MDR Connector**		**cM44 & cM6x Baseboard SCSI-2 50-Pin Connector**	**M44, M6x, & SBC6x Baseboard IDC50 Connector**	**M44 & M6x Baseboard DB15 Connector**	**Function**
	Module Site 0	**Module Site 1**				
1..35	34..50, 83..100	9..25, 58..75	1..25, 41..50	1..35		Reserved
36	33	8	40	36	1	AGND
37	82	57	39	37	9	Channel 1 Input
38	32	7	38	38	2	AGND
39	81	56	37	39	10	Channel 2 Input
40	31	6	36	40	3	AGND
41	80	55	35	41	11	Channel 3 Input
42	30	5	34	42	4	AGND
43	79	54	33	43	12	Channel 4 Input
44	29	4	32	44	5	AGND
45	78	53	31	45	13	AGND
46	28	3	30	46	6	AGND
47	77	52	29	47	14	External Gate
48	27	2	28	48	7	AGND
49	76	51	27	49	15	AGND
50	26	1	26	50	8	Sync Output

TABLE 46. AIX I/O Connector Pinout

AIX20 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50-Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1..35	34..50, 83..100	9..25, 58..75	1..25, 41..50	1..35		Reserved
36	33	8	40	36	1	AGND
37	82	57	39	37	9	Channel 1 Input
38	32	7	38	38	2	AGND
39	81	56	37	39	10	Channel 2 Input
40	31	6	36	40	3	AGND
41	80	55	35	41	11	Channel 3 Input
42	30	5	34	42	4	AGND
43	79	54	33	43	12	Channel 4 Input
44	29	4	32	44	5	AGND
45	78	53	31	45	13	AGND
46	28	3	30	46	6	AGND
47	77	52	29	47	14	External Gate
48	27	2	28	48	7	AGND
49	76	51	27	49	15	AGND
50	26	1	26	50	8	External Clock Input

TABLE 47. **AIX20 I/O Connector Pinout**

Functions

Analog Input

The input to the AIX (AIX20) has a 50-ohm impedance to ground allowing the user to use shielded cable to preserve signal integrity and maintain minimal cross talk between channels. The input buffering is done with wide band, low distortion amplifiers maintaining the low signal to noise and distortion of the A/Ds. The AIX (AIX20) module inverts in the input signal, a 2 volt signal in would read -2 volts out. This inversion is software corrected. The A/Ds are stacked in pairs on the data bus to allow two channels to be read in one cycle and have an input range of 4 Volts peak to peak. For more in depth information on the AD9260 refer to the data sheet, which can be found on Analog Devices website (www.analog.com).

AIX Only: The analog to digital converters (AD9260) is a 16 bit device with a maximum output rate of 2.5 MHz capable of instrumentation grade conversion. The AD9620 achieves its high dynamic range by oversampling the input 8:1. There is also a digital decimation filter at the nyquist frequency preventing any aliasing or undersampling.

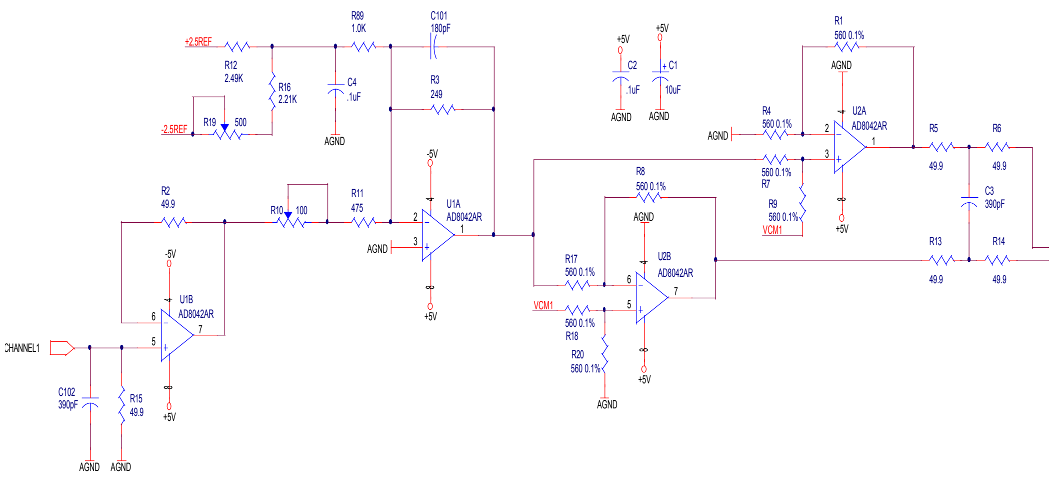


FIGURE 41. **AIX Input Schematic**

AIX20 Only: The AIX20 is basically identical to the AIX Module except for the A/D is connected in the 1x decimation mode turning it into a 12 bit 20 MHz A/D with no anti-aliasing filters. All other features are identical to the AIX module. The capacitors on the schematic have been changed to accommodate the increased input bandwidth of the A/D and the new values are shown below.

Capacitor	Value
C102	0pF
C101	47pF
C3	82pF

TABLE 48. **AIX20 Input Filtering**

Analog Input Conversion Trigger Clock: AIX20 module

A/D data acquisition on the AIX20 module may be triggered by either the OMNIBUS host board DDS signal or by an external signal source. The clock source is selected by writing to the A/D clock selection register: setting bit 0 selects the DDS signal as the conversion trigger, while setting bit 1 selects an the external source.

The DDS signal is provided by the host and configured using host-specific software: see the documentation for the OMNIBUS host card in use for details on setting the DDS frequency. The external clock signal is TTL standard and must be connected to the External Clock Input pin (see the pinout definitions above).

Analog Input Conversion Trigger Clock: AIX module

A/D data acquisition on the AIX module is triggered by a sample clock provided by the OMNIBUS host on the DDS signal pin. Since the A/D converters used on the AIX module are over-sampled by a factor of 8:1, the DDS generator on the OMNIBUS host must be programmed to run 8 times faster than the desired sample rate. For example, to acquire data at 2.5 MHz, the DDS would be programmed for a 20 MHz clock frequency.

There is no provision on the AIX module for connecting an external clock as the conversion rate source.

Analog Input Trimming

The ADC offsets and gain are set at the factory, but may be trimmed by the user with the following procedure below. Errors should be trimmed at the expected normal operating temperature. The trim pot locations are given in table below.

1. Short the inputs for each channel to AGND on connector P1. You may want to do this out at your sensor to include any error sources present in the application circuitry. Trim offset pots such that a zero code is observed when continuously sampling the converters.

2. Connect one input to a voltage reference and check gain for proper codes. Full scale inputs, at a gain of 1, are +/-2V and should read raw ADC codes of 32767 and -32767 respectively.

Channel	Gain	Offset
1	R10	R19
2	R31	R40
3	R52	R61
4	R73	R82

TABLE 49. **AIX (AIX20) Gain and Offset Adjustment**

CHAPTER 8

DAC40 Module

Module Introduction

The DAC40 OMNIBUS module gives the target processor card four channels of very high speed 40 MHZ, 14-bit resolution digital input to analog output conversion (D/A). This makes the DAC40 module well suitable for use in high-speed signal and arbitrary waveform generation, communications, and control systems. The DAC40 module uses four Analog Devices AD9764 D/As (one for each channel) along with the necessary output buffering, allowing for independent channel control. The AD9464 features 72 dB total harmonic distortion and a spurious free dynamic range of 75 dB, which is ideal for communications applications. The output is +/- 1V into a 50-ohm load, which is DC accurate and can be calibrated.

The DAC40 module has special memories (SARAM) placed between the DSP and the D/As. This allows the module to load waveform tables into memory for uninterrupted repeated playback independent of the DSP. This relieves the DSP from servicing the D/A at the high rates generally required for waveform generation. The DSP simply loads and playback table into RAM after which it may be played back continuously at full rate. The DSP may update the contents of the SARAM while playback is occurring without interrupting the current playback. Additionally, the DSP may bypass the SARAM and communicate directly with the DAC when necessary.

Software support for the DAC40 is provided with Innovative Integration's Zuma software development package. Example programs illustrate the library functions used with the DAC40 and speed up application development.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

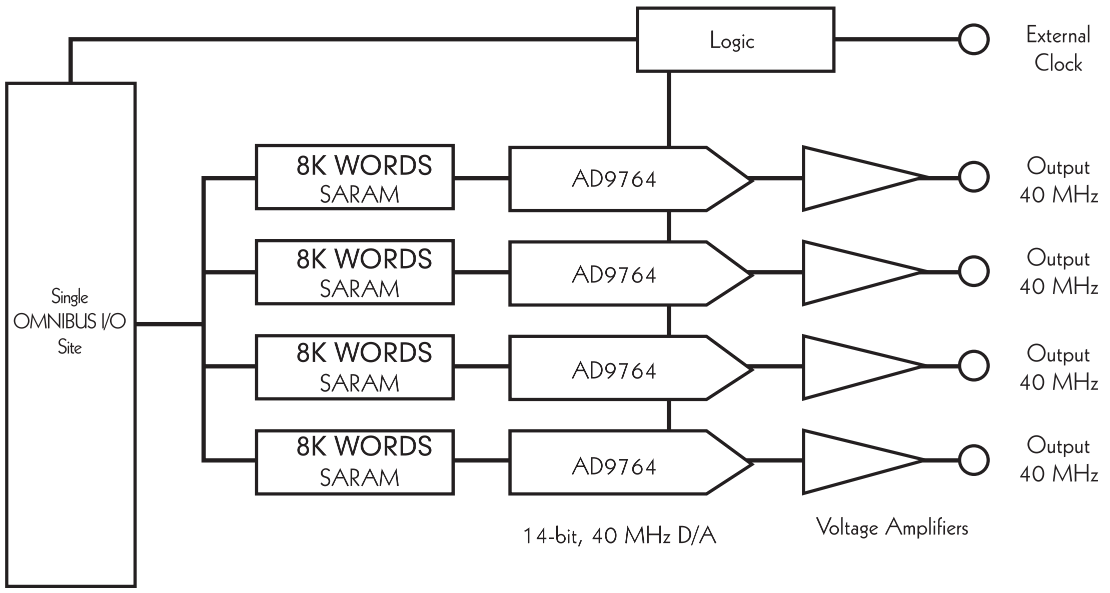


FIGURE 42. **DAC40 Block Diagram**

Bus Type:	Compatible with all I.I. products; 32-bit. Consumes one interrupt to host DSP. Wait-state depends on host platform	**Max. Update Rate:**	40 MHz
Physicals:	OMNIBUS mezzanine card 2.0” X 4.6”	**Settling Time:**	35 nS
D/A Converters:	Four Analog Devices AD9764	**Glitch Impulse:**	5 pV-s
Resolution:	14-bit	**SFDR:**	73 dB
Output Rate:	+/- 1 V which is DC accurate and may be calibrated.	**THD**	-72 dB
Dynamic Range:	75 dB	**SNR:**	64 dB
Offset Error:	Trimmable	**DNL:**	-1.0/+1.25 LSB
Gain Error:	Trimmable	**INL:**	+/- 2.25 LSB
Filter Cutoff:	N/A	**Clock**	Software selectable: Internal/External
Conversion Trigger:	Software programmable, internal or external.	**Max. Write Rate**	50 MHz (20 nS cycle) Random Access.
Interface to DSP:	High performance SARAM.		

The target Peripheral Library provides support for each of the DAC40 functions, and the following sections give descriptions of the support routines. Refer to the **dac40.h** or **dac40ing.h** files located in the **`c:/<target board>/Include/Target/Analog`** director and **Omnibus.hlp or Zuma.hlp** Windows helpfile for complete details on each function.

Function Name	**Operation**
timebase()	Sets sample rate of D/As using DDS.
DAC40_initialize()	Resets the DAC40 module.
DAC40_load_symbol()	Updates SARAM memory for specified D/A pair from local buffer, using DMA channel.
DAC40_activeate_symbol()	Activate specified waveforms in SARAMs.
DAC40_trigger_dac()	Selects clock source used to trigger conversions for all D/A channels.
DAC40_gate()	Globally enables/disables (gates) acquisitions to all D/A channels
DAC40_reset()	Resets the DAC40 SARAM devices.
DAC40_set_interrupt_mode()	Set SARAM interrupt signaling mode.
DAC40_read_status()	Read back of SARAM interrupt flags
DAC40_clear_buffer_flags()	Clear SARAM interrupt flags
DAC40_correct_dac()	Convert twos-complement sample data to internal format used by D/A converter channel
DAC40_correct_dac_pair()	Convert AIO14 structure D/A sample pair to internal D/A format
DAC40_write_idrom()	Write module identification information.
DAC40_read_idrom()	Read module identification information.
DAC40_fast_omnibus()	Programs Omnibus for fast accesses. (For M6x only)
DAC40_normal_omnibus()	Programs Omnibus for normal accesses. (For M6x only)

TABLE 50. C Language DAC40 Functions

The DAC40 library functions can be divided into several groups:

1. Sample rate control
2. D/A control
3. SARAM control
4. Identification readback

Sample Rate Control

The timebase used to trigger D/A conversions on the DAC40 may be precisely controlled via the baseboards 9850 DDS timer. The **timebase()** function controls the sampling rate used by all the A/D converters within the application. The DDS clock must be set up to run at one times the desired sample rate.

D/A Control

The D/As on the DAC40 begin sampling data after the trigger clock begins running. The FPGA logic on the board controls the gating of data out of the SARAMs.

SARAM Control

Each 14-bit D/A converter on the DAC40 is interfaced to the baseboard via a 16-bit SARAM. The DAC40 module is equipped with four SARAMs. They are decoded onto the bus as two pairs of 8Kx32-bit buffers, which are treated by the DAC40 driver as *symbol* memory. A symbol is a waveform, up to 4Ksamples in length, which may be dynamically loaded by application software. Because there are two symbol buffers, it is possible to simultaneously play a waveform from one of the buffers while loading another. This operation may be performed within interrupt routines in order to generate dynamic waveforms of arbitrary length in response to end-of-buffer conditions detected by the SARAM.

To begin playing a sampled data waveform from the SARAMs, a clock must be routed to the D/As to act as a trigger source using the **DAC40_trigger_dac()** function. This clock must be routed to the SARAMs to allow the synchronous-interface of the SARAMs to clock data out to the D/As, using the **DAC40_gate()**function. The gate function allows data to begin clocking out of all four SARAMs simultaneously. The SARAMs will continue to provide data to the D/As until they reach their pre-programmed buffer boundaries. Then they optionally signal an interrupt to the baseboard and continue playing the waveform from the beginning of the “other” internal symbol buffers.

If enabled at the processor level, the SARAMs interrupt (flag) the processor when the synchronous SARAM interface reaches the end of buffer 1 or the end of buffer 2. The state of these SARAM buffer flags can be read at any time with the **DAC40_read_status()** function. The driver software places the SARAMs into the “buffer-chain” operational mode, to facilitate continuous, re-programmable waveforms.

To write waveform data to the SARAM buffers, use the **DAC40_load_symbol()** function, which takes the pair number to read as an argument (pair 0 for channels 0 & 1, pair 1 for channels 2 & 3), the module site number (0..2), the symbol number (0..1), the address of the waveform in baseboard memory to be copied to the SARAM and the size of the waveform. The data comes in stacked on the 32-bit bus with bits 0..13 being the first channel of the pair and bits 16..30 being the second channel of the pair. This function uses the baseboard DMA channel 0 to perform the move.

Once the waveforms are located into symbol memory within the SARAM, use the **DAC40_activate_symbol()** function to cause either of the two internal buffer pointer registers located within the SARAM to point to the waveform to be played out. This function allows independent control of both of the internal buffer pointer registers; the `symbol` parameter in this function corresponds to the `symbol` parameter in the previous **DAC40_load_symbol()** function call. The `idx` parameter specifies which internal buffer pair is to be modified.

To provide continuous, dynamic waveform playback, follow this sequence. Load the first waveform into symbol 0 memory. Load the second waveform into symbol 1 memory. Then, activate buffer pair 0 to point to the first waveform and buffer pair 1 to point to the second symbol. Gate the D/As on, allowing playback to commence. When the end-of-buffer 1 condition is reached, the SARAMs will interrupt the CPU and they will automatically begin playing the second waveform symbol. Your ISR should then load a new symbol into symbol buffer 0 memory and the buffer pair 0 should be activated to point to this new waveform. When the end-of-buffer 2 condition is later reached, you'll need to load a new symbol into symbol buffer 1 memory and buffer pair 1 should be activated to point to this new waveform. Meanwhile, the SARAMs will be playing the contents of buffer 0. This process may be repeated *ad-infinitum*.

The SARAMs may be initialized or reset with the **DAC40_reset()** function.

Identification Readback

The **DAC40_read_idrom()** function is used to read the identification ROM on the DAC40 in order to check its identity and revision level. The function fills out a DAC40`_ID` structure with the information stored in the ID ROM on the module. The DAC40`_ID` structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string "DAC40")
2. Module revision level (single character revision level, i.e. 'A')
3. Checksum

This data is preprogrammed at the factory and should not be altered by the user.

Memory Mapping

The DAC40 module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** file included with the host board Development Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	R/W	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address	Value
D/A CLK Source Selection	W	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0	0 – DDS CLK* 1 – External CLK 2 – On Board Xtal 3 – TMR0
Simultaneous Enable (Gate)	W	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1	0 – Playback Disabled* 1 – Playback Enabled
D/A & SARAM Reset	W	IOMOD0 + 0x2	IOMOD4 + 0x2	IOMOD8 + 0x2	Any Value – Reset
Pair 0 Address MSB Control	W	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3	0 – Access to lower 4096 words* 1 – Access to upper 4096 words
Pair 1 Address MSB Control	W	IOMOD0 + 0x4	IOMOD4 + 0x4	IOMOD8 + 0x4	0 – Access to lower 4096 words* 1 – Access to upper 4096 words
SARAM Start Address Buffer # 1	R/W	IOMOD0 + 0x8	IOMOD4 + 0x8	IOMOD8 + 0x8	
SARAM End Address Buffer # 1	R/W	IOMOD0 + 0x9	IOMOD4 + 0x9	IOMOD8 + 0x9	
SARAM Start Address Buffer # 2	R/W	IOMOD0 + 0xA	IOMOD4 + 0xA	IOMOD8 + 0xA	
SARAM End Address Buffer # 2	R/W	IOMOD0 + 0xB	IOMOD4 + 0xB	IOMOD8 + 0xB	
SARAM Flow Control Register	R/W	IOMOD0 + 0xC	IOMOD4 + 0xC	IOMOD8 + 0xC	
SARAM Status Flag Clear	W	IOMOD0 + 0xD	IOMOD4 + 0xD	IOMOD8 + 0xD	0 – Clears Flags for Buffers 1 & 2 1 – Clears Flag for Buffer # 2 2 – Clears Flag for Buffer # 1
SARAM Status Flag Read	R	IOMOD0 + 0xD	IOMOD4 + 0xD	IOMOD8 + 0xD	
SARAM Pair 0 Memory	R/W	IOMOD1	IOMOD5	IOMOD9	
SARAM Pair 1 Memory	R/W	IOMOD2	IOMOD6	IOMOD10	
IDROM SDA	R/W	IOMOD3 + 0x0	IOMOD7 + 0x0	IOMOD11 + 0x0	
IDROM SCK	W	IOMOD3 + 0x0	IOMOD7 + 0x0	IOMOD11 + 0x0	

TABLE 51. **DAC40 Memory Map (* Default at Reset)**

Interrupt Usage

Processor interrupts are generated from the DAC40 when the sequential pointer reaches the End Of Buffer pointer for either buffer 1 or 2. The interrupt must be cleared by writing a zero to the appropriate bit location (see table above) in the SARAM Status Clear Flag Register to allow further interrupts to occur.

Sequential playback of a waveform will still occur without the use of interrupts or flags, but the number of times a waveform has been played out will be unknown.

Pin Connector I/O

The DAC40 output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* for additional details on how to connect to the DAC40's I/O pins.

DAC40 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50-Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1..35	34..50, 83..100	9..25, 58..75	1..25, 41..50	1..35		Reserved
36	33	8	40	36	1	Reserved
37	82	57	39	37	9	Reserved
38	32	7	38	38	2	Reserved
39	81	56	37	39	10	Reserved
40	31	6	36	40	3	Reserved
41	80	55	35	41	11	Reserved
42	30	5	34	42	4	Reserved
43	79	54	33	43	12	Reserved
44	29	4	32	44	5	Reserved
45	78	53	31	45	13	Reserved
46	28	3	30	46	6	Reserved
47	77	52	29	47	14	Reserved
48	27	2	28	48	7	Reserved
49	76	51	27	49	15	AGND
50	26	1	26	50	8	EXT_TRIG_EN

TABLE 52. DAC40 I/O Connector Pinout

I/O Connector	Function
J1	Analog Output Channel 0
J2	Analog Output Channel 1
J3	Analog Output Channel 2
J4	Analog Output Channel 3
J5	EXT_CLK_INPUT

TABLE 53. DAC40 MCX I/O Connector Pinout

Functions

Analog Output

The DACs are Analog Devices AD9764 high-performance 14-bit 100 MHz low-power CMOS converters. The differential current output of the DACs is converted to a single-ended output using low distortion op-amps and is made available to the user via 50-ohm MCX connectors. The range of the output amp is +/-1.0 volts into a 50-ohm load, and +/-2 volts into high impedance. Positive full-scale output is achieved by writing 16384 while negative full-scale is achieved by writing 0.

Between the DACs and the processor are four sequential access RAMs (SARAMs) capable of outputting data at up to 40 MHz. A SARAM is a dual port memory with one port being a random access port (DSP side) and the other port being a sequential access port (DAC side). Each SARAM can be configured to playback up to 8K samples repeatedly or be used as two ping-pong buffers where one half may be playing out while the other half is being written.

Digital data is written to the SARAMs for output via a set of memory mapped locations. The data bus is connected to the SARAMs in pairs. The following table gives the channel assignments and the bit fields used to write to each SARAM.

Module Register	**Data Bus Bit Field**	**D/A Channel**
SARAM Pair 0 Data Write	D0..D13	0
	D16..D29	1
SARAM Pair 1 Data Write	D0..D13	2
	D16..D29	3

TABLE 54. **DAC40 Channel Write Addresses and Bit Fields**

The data written to the D/A is in straight binary format.

The SARAM memories are paired along with the D/A converters and data is clocked out of the sequential access port at the D/A update rate. It should be noted that each pair of SARAMs is mapped to the OMNIBUS interface at a single address. Therefore, it is not possible to store data to an individual channel's sample memory at a time: rather, data is clocked into the pair's SARAM buffers simultaneously. For example, a write to the SARAM pair 0 memory causes data to be stored for both D/A channels 0 and 1 simultaneously.

Also, note that the total SARAM buffer space (8 Ksamples) is larger than the size of a single OMNIBUS IOMODx address space. Each SARAM must be addressed as paged memory where one half of the SARAM storage is addressable at a time. The paging is controlled by the Pair 0 and Pair 1 MSB Control Registers: writing zero to each register enables access to the lower 4Ksamples of the SARAM, while writing one enables access to the upper 4Ksample region.

Analog Output Conversion Triggering

The D/As may be triggered by a signal from one of the following four sources:

1. The host hoard's DDS signal source (0 – 25 MHz frequency range).
2. An external clock source.
3. The on board crystal oscillator (optional).

4. The host board's TMR0 signal (max frequency varies depending on host board).

The DAC40 trigger source is controlled by the D/A Clock Source Selection register. By programming the values given in the memory map software, you may select one of the four available sources.

DDS Trigger Source

Selecting the DDS clock source as the trigger signal allows host software to directly control the DAC40 conversion rate by programming the DDS synthesizer frequency. This option limits the DAC40 sampling frequency to the range available with the DDS source (typically 0-25 MHz, varies by OMNIBUS host board). The DDS source on the OMNIBUS host must be programmed to the conversion frequency desired for the DAC40.

External Clock Source

The DAC40 can receive a TTL compatible clock signal via connector (J5). This allows the external hardware to directly drive the conversion rate of the module. Connector (J5) is an MCX compatible female right angle plug, which accepts a single MCX in-line jack. The connector signal conductor is terminated to ground with a 50-ohm resistor (R65). The shell is tied to digital ground on the module. Recommended connectors are AMP 829550-2 terminating RG178 cable.

Hardware selected to drive the (J5) connector should be capable of delivering a TTL compatible signal source to the 50-ohm terminated input. Loading is a single CMOS load. Clock jitter should be kept to a minimum (especially at high conversion rates). Refer to the AD9764 data sheet for specifications on clock phase and jitter for the converters.

On-Board Crystal Oscillator

The DAC40 module accepts a standard half-size TTL oscillator in a through-hole package as a custom fixed frequency sample rate timebase. If this option is selected then the output of the oscillator becomes the D/A update timebase. The DAC40 is compatible with oscillators from CTX (MXO-45 series), Epson (SG-531 series), and Ecliptek (EC1100 series). The oscillator is mounted directly to the DAC40 module at location (U19).

Host Board TMR Signal

Similar to the DDS clocking option, the DAC40 may also use the OMNIBUS TMR0 signal to trigger updates on the D/A converters. This signal is driven by host board timer hardware and exact capabilities vary by host card. See the Host card hardware manual for details on OMNIBUS timer options.

Analog Output Filtering

C1, C16 ,C31 and C46 provide filtering with a simple single pole roll-off across a 25-ohm load. The factory setting for this roll-off is 220 pF (~28 MHz).

The following figures give the output buffer schematics for each channel.

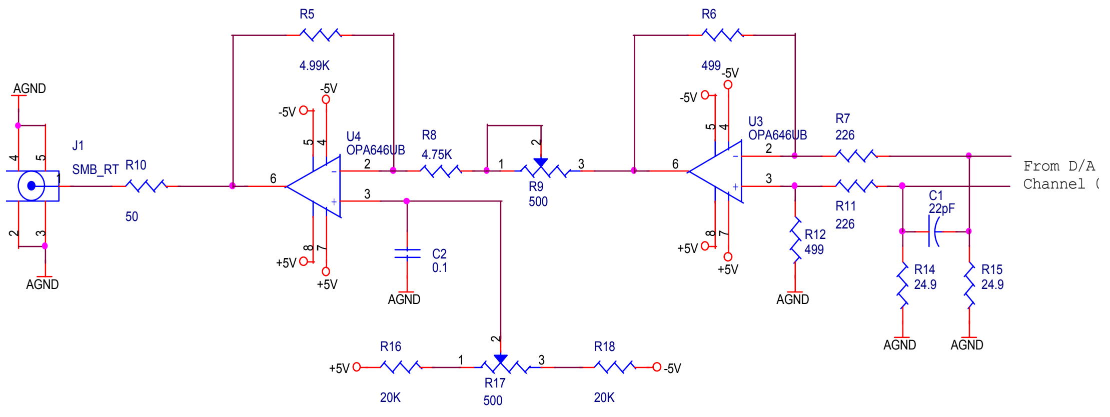


FIGURE 43. **DAC40 Channel 0 Output Buffer**

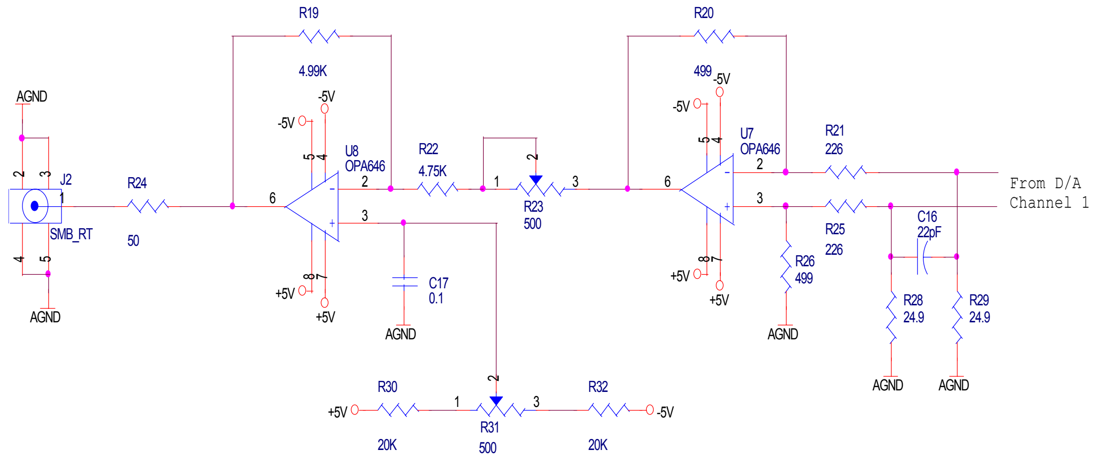


FIGURE 44. **DAC40 Channel 1 Output Buffer**

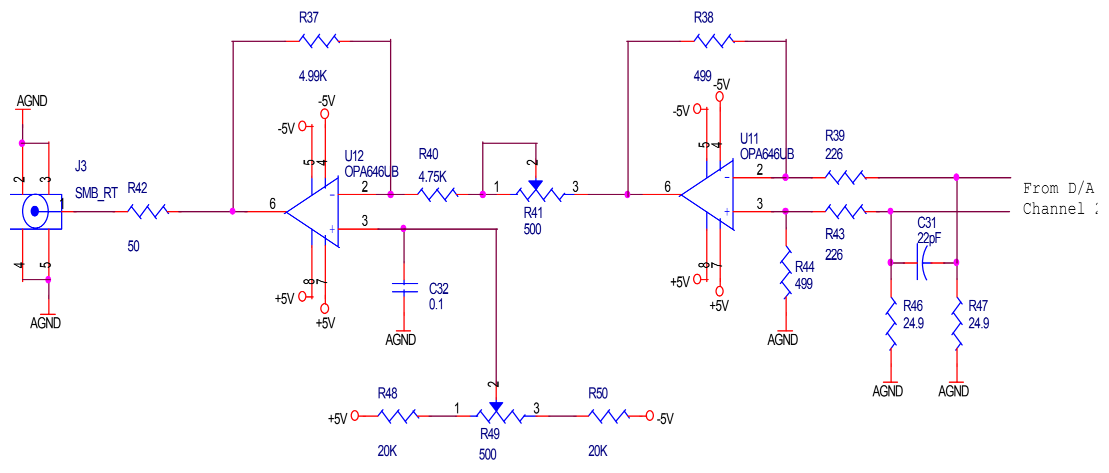


FIGURE 45. DAC40 Channel 2 Output Buffer

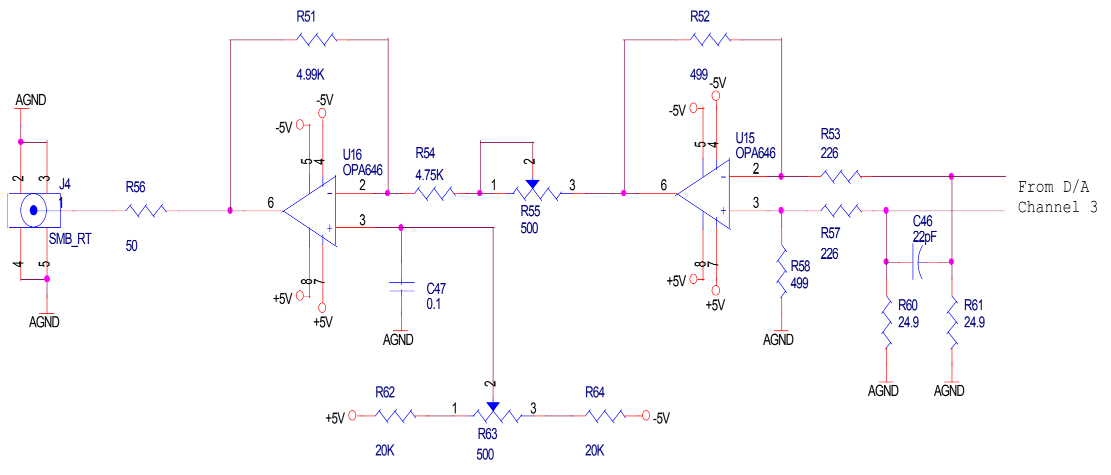


FIGURE 46. DAC40 Channel 3 Output Buffer

D/A Output Trimming

The DAC40 supports trim on each D/A output for both gain and offset. The following table gives the reference designators for the trim potentiometers.

D/A Channel	Offset Trim Pot	Gain Trim Pot
0	R17	R9
1	R31	R23
2	R49	R41
3	R63	R55

TABLE 55. **DAC40 Offset and Gain Adjustment Potentiometers**

CHAPTER 9 DIG Module

Module Introduction

The DIG module provides the target card processor with 32 bits of additional bi-directional digital bit I/O. The module is also equipped with dual channel USART serial ports driven by a optional RS232 or RS422 line driver/receivers. The board is configured to the type of driver/receiver selected. These two serial ports provide an easy interface to a wide variety of measurements and process control devices. This module is commonly used for sensing digital inputs and as control bits for industrial processing equipment.

The module's digital I/O can drive opto-couplers and industrial interface modules that allow the DSP to control high-current, high-voltage loads as part of a control or measurement system. In addition, the digital I/O port is used as a high-speed communications port, achieving speeds of up to 40 Mbytes per second.

The two serial ports can be configured in a variety of data formats for both asynchronous and synchronous communications. Each serial port may be used at up to 10 Mbits per second in synchronous mode, or up to 32 kbaud in asynchronous mode. The serial ports allow for the DSP card to communicate a control over instruments, act as a slave to a host processor, or work together in a multi-drop configuration for distributed processing.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

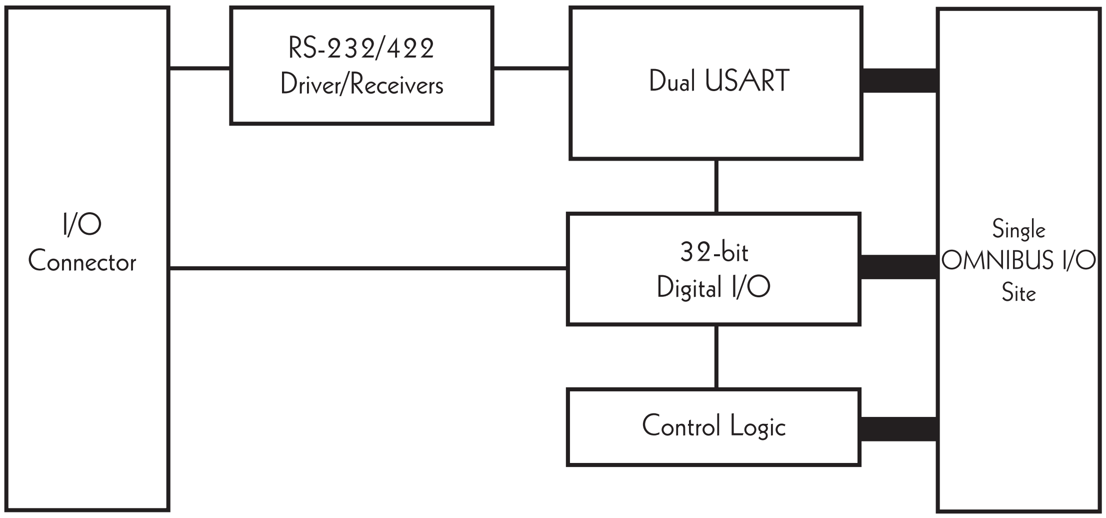


FIGURE 47. **DIG Block Diagram**

Bus Type:	Compatible with all I.I products; 32-bit. Consumes one interrupt to host DSP. Wait-state depends on host platform.	**Interface to DSP:**	Memory-mapped
Power Requirements:	5 V @ 150 mA (no load on outputs); +15 V @ 25 mA; -15 V @ 25 mA	**Serial I/O:**	Two independent channels. (Philips 26C562)
Physicals:	OMNIBUS mezzanine card 2.0” X 4.6”	**Serial Protocols:**	Asynchronous: 5 - 8 bits, optional parity. Up to 32k baud.
Digital I/O:	32-bit total.	**Synchronous:**	Sync., bisync, SDLC, etc. Up to 10 Mbit/sec.
Direction Control:	Configurable byte-by-byte for input or output	**Data FIFO:**	16 character FIFOs on each channel.
Drive Capability:	Drive capability 32 mA source / 64 mA sink. Can directly drive opto-couplers and LED’s.	**Line Interfaces:**	RS232; RS422 Factory Configured
Data Triggering:	Input latch on DSP read or external TTL clock. Outputs latched on DSP write.	**Handshaking:**	RTS/CTS hardware handshaking. Jumper Configurable.
Access Speed:	Capable of 7.5 MHz operation. Same as I/O module bus rate.	**Interface to DSP:**	Memory-mapped.

The target Peripheral Library provides support for each of the DIG functions, and the following sections give descriptions of the support routines. Refer to the **dig.h** or **dio.h** file located in the `c:/<target board>/Include/Target/Digital` director and the **Omnibus.hlp or Zuma.hlp** Windows help file for complete information on each function.

Function Name	Operation
DIO_ser_init()	Initialize the USART for communications.
DIO_rx()	Get a character from the serial interrupt receive buffer.
DIO_poll_ser_tx()	Polled serial character transmit.
DIO_poll_ser_rx()	Polled serial character receive.
DIO_poll_ser_chk_tx()	Checks for available space in the transmit FIFO.
DIO_poll_ser_chk_rx()	Checks for receipt of character in the receive buffer.
DIO_dig_dir()	Initialize the direction of the digital I/O port.
DIO_read_dig()	Read the digital I/O port.
DIO_write_dig()	Write the digital I/O port.
DIO_install_serial_int_vector()	Installs the interrupt handler.
DIO_deinstall_serial_int_vector()	Uninstalls the interrupt handler.
DIO_read_idrom()	Read identification information.
DIO_write_idrom()	Write identification information.

TABLE 56. **C Language DIG Functions**

The DIG library functions can be divided into several groups:

1. Digital I/O programming
2. Serial port support
3. Identification readback

Digital I/O Programming

Digital I/O programming on the DIG module is very similar to interaction with the baseboard digital I/O port. Three functions are provided to control the port and read/write data. **DIO_dig_dir()** sets the direction control bits for each byte in the 32-bit port, while **DIO_read_dig()** and **DIO_write_dig()** read and write data from/to the port. When reading the port, remember that only the bytes programmed for input will return data from the external pins: pins programmed for output will return the data last written to the pins. Similarly, when writing to the port, only the bytes programmed for output will be driven to the I/O pins by the DIG module.

Serial Port Support

The two channel USART serial port device on the DIG module is supported by routines for polled transmission and polled and interrupt-driven reception (interrupt-driven transmission is not supported by the Peripheral Library). Before the serial port can receive or transmit characters, it must be initialized using the **DIO_ser_init()** function. Each channel on the device is initialized using separate calls to the **DIO_ser_init()** function with different arguments. The standard drivers support communication rates from 50 to 38400 baud in both the interrupt and polled modes.

Once initialized, data may be received and transmitted by calling the three data movement functions, **DIO_ser_poll_rx()**, **DIO_ser_poll_tx()**, and **DIO_rx()**. In polled serial port mode, the **DIO_ser_poll_tx()** function writes a value to the transmit serial port register of the appropriate USART channel and waits for the transmission to complete. The **DIO_ser_poll_rx()** function waits for a character to become available in the receive buffer of the USART channel and retrieves the character once it becomes available.

In interrupt driven mode, the **DIO_ser_poll_tx()** function is still used to transmit characters (since interrupt driven transmission is not supported). Character reception occurs as a result of the serial port interrupt handler, serial receive buffer, and the **DIO_rx()** function. The serial port interrupt routine is installed and enabled by the **DIO_ser_init()** routine if interrupt driven reception is enabled. Once installed the interrupt routine will be called by a hardware interrupt from the DIG module whenever a character becomes available in the receive FIFO of the USART. The interrupt routine empties all available characters from the FIFO and places them in the serial receive buffer, where they are available for pickup by the **DIO_rx()** routine. The **DIO_rx()** routine waits for an available character in the serial receive buffer, reads one when available and returns the value to the calling routine.

Note: interrupts MUST be globally enabled and the host processor's interrupt input must be connected to the interrupt output of the DIG module for interrupt-driven serial reception to function. The **DIO_ser_init()** function installs the interrupt vector and locally enables the interrupt on the processor, but does not affect the global interrupt enable in the processor status register. This bit must be turned on by a call to **enable_interrupts()** (see interrupt discussion above) for characters to be received. The DIG's interrupt output (presented on target external interrupt zero if the module is installed in I/O slot 0, or target external interrupt two if the module is installed in I/O slot one) must be connected via the target's processor interrupt jumper header to the interrupt input specified in the **DIO_ser_init()** call (see the *target card Manual, "Hardware*" section for information on configuring the target interrupt jumper header).

By default, the **DIO_ser_init()** routine sets the USART up for an 8N1 non-return-to-zero serial port protocol. If RS232 drivers are populated on the DIG module, this protocol will be logically and electrically compatible with common personal computer, modem, and "dumb" terminal serial ports. Other protocols are possible, including high-speed synchronous and other forms of asynchronous protocols. Consult the source code for the **DIO_ser_init()** routine as well as the target card manual for information on the other protocols supported by the USART.

Identification Readback

The **DIO_read_idrom()** function is used to read the identification ROM on the DIO to check its identity and revision level. The function fills out an DIO_id structure with the information stored in the ID ROM on the module. The `DIO_id` structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string “DIO”).
2. Module revision level (single character revision level).
3. Serial port A channel driver type (RS232 or RS422).
4. Serial port B channel driver type (RS232 or RS422).
5. Checksum.

This data is preprogrammed at the factory and should not be altered by the user.

Memory Mapping

The DIG module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** file included with the Development Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
UART	IOMOD0..IOMOD0+0x3f	IOMOD4..IOMOD4+0x3f	IOMOD8..IOMOD8+0x3f
Digital I/O Data	IOMOD1	IOMOD5	IOMOD9
Digital I/O Byte 0 Direction Control	IOMOD2 + 0	IOMOD6 + 0	IOMOD10 + 0
Digital I/O Byte 1 Direction Control	IOMOD2 + 1	IOMOD6 + 1	IOMOD10 + 1
Digital I/O Byte 2 Direction Control	IOMOD2 + 2	IOMOD6 + 2	IOMOD10 + 2
Digital I/O Byte 3 Direction Control	IOMOD2 + 3	IOMOD6 + 3	IOMOD10 + 3
ID ROM Data	IOMOD3	IOMOD7	IOMOD11

TABLE 57. **DIG Memory Map**

Interrupt Usage

The DIG module's USART is capable of generating interrupts to the host processor on the module slot's external interrupt line (external interrupt 0 or 1 for slots 0 and 1, respectively). The Development Package libraries contain support for interrupt driven data communications via this interrupt input: if the application software expects to be able to receive interrupts from the USART, the appropriate jumper must be set (or the source programmed, depending on host board) on the host's processor interrupt header.

Pin Connector I/O

The DIG output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Note that two different pinouts are given for the UART drivers functions: the pinout functions of a particular DIG module vary depending on the line transceiver configuration requested on the order. Refer to the host card's *Instruction Manual*, *"Hardware"* for additional details on how to connect to the DIG's I/O pins.

DIG Omnibus M173280-3 Connector Pin Number	**ChicoPlus & Hombre Baseboard 100 Pin MDR Connector**		**cM44 & cM6x Baseboard SCSI-2 50-Pin Connector**	**M44, M6x, & SBC6x Baseboard IDC50 Connector**	**M44 & M6x Baseboard DB15 Connector**	**Function (RS232 Serial)**	**Function (RS422 Serial)**
	Module Site 0	**Module Site 1**					
1..32	35..50, 85..100	10..25, 60-75	1..25, 44..50	1..32		Digital I/O bits	Digital I/O bits
33	84	59	43	33		External Readback clock (active low)	External Readback clock (active low)
34	34	9	42	34		Reserved	Rx Clock/CTS B-
35	83	58	41	35		Rx Clock/CTS B	Rx Clock/CTS B+
36	33	8	40	36	1	Reserved	Rx Data B-
37	82	57	39	37	9	Rx Data B	'+-+AZ
38	32	7	38	38	2	Reserved	Tx Clock/RTS B-
39	81	56	37	39	10	Tx Clock/RTS B	Tx Clock/RTS B+
40	31	6	36	40	3	Reserved	Tx Data B-
41	80	55	35	41	11	Tx Data B	Tx Data B+
42	30	5	34	42	4	Reserved	Rx Clock/CTS A-
43	79	54	33	43	12	Rx Clock/CTS A	Rx Clock/CTS A+
44	29	4	32	44	5	Reserved	Rx Data A-
45	78	53	31	45	13	Rx Data A	Rx Data A+
46	28	3	30	46	6	Reserved	Tx Clock/RTS A-
47	77	52	29	47	14	Tx Clock/RTS A	Tx Clock/RTS A+
48	27	2	28	48	7	Reserved	Tx Data A-
49	76	51	27	49	15	Tx Data A	Tx Data A+
50	26	1	26	50	8	GND	GND

TABLE 58. DIG I/O Connector Pinout

Functions

Digital I/O

The DIG module provides 32 bits of bi-directional, software programmable digital I/O. Each byte of the 32-bit port is direction programmable via a hardware register and input data may be latched either by a software read or an external active low readback strobe.

The direction control registers provide for software control of the drive direction of the port. Each byte is individually controllable by writing a zero (to select output) or a one (to select input) to the respective direction control register.

The data register allows software to directly read data from port pins programmed for input, or write data to pins programmed for output.

Since the digital I/O port is a latching port (meaning that inputs are clocked in on the edge of a strobe signal and held for the processor to read), there is a jumper selection, which allows the user to pick the strobe source. Jumper (JP1) allows the user to select either software read clocking (pins 2-3 shorted) or external hardware clocking (pins 1-2 shorted). If software clocking is selected, then the port latches programmed for input will clock in the digital data present on the external pins at the beginning of a read cycle executed on the port. (30-50 ns before the data is returned to the processor, depending on processor clock speed). If external clocking is selected, then the port will latch data on the falling edge of the TTL signal EXT_DIG_RD_CLK* on the module's I/O connector (pin 33). The data will be held for the processor to read until the next low-going edge of the EXT_DIG_RD_CLK* signal.

UART

The DIG module includes a two channel USART for use in systems requiring standard serial communications. The Phillips (SC26C562) device is used to provide programmable FIFO-buffered asynchronous or synchronous transceiver capabilities. The DIG module's design includes an option for either RS232 or RS422 transmitters/receivers, and can support external clocking in synchronous mode.

The host board Development Package includes drivers for the SC26C562 which support character-based polled or interrupt driven data communications. See the host's *Development Package Manual* for more information on the supplied serial drivers, as well as example code and libraries for using the device.

The SC26C562 device uses a set of 64 byte-wide registers to control its functionality. These registers are mapped onto the OMNIBUS in the least significant eight bits of the host processor's data bus. The upper 24 bits are ignored by the CS26C562, and may be left to float on writes, but should be software masked on reads. The SC26C562 is mapped into the first 64 addresses starting at IOMOD0 in slot 0, at IOMOD4 in slot 1, or at IOMOD8 in slot 2. The register set will alias every 64 locations (i.e. CS26C562 register 0 appears at IOMOD0, IOMOD0+64, IOMOD0+128, etc.).

Several jumpers on the DIG module control the use of handshaking and clock functions, as detailed in the following table below.

Jumper	Setting	I/O Pin Function
JP5	1-2	Pin 47 = Ch. A Tx Clock
	2-3	Pin 47 = Ch. A RTS
JP6	1-2	Pin 43 = Ch. A Rx Clock
	2-3	Pin 43 = Ch. A CTS
JP7	1-2	Pin 39 = Ch. B Tx Clock
	2-3	Pin 39 = Ch. B RTS
JP8	1-2	Pin 35 = Ch. B Rx Clock
	2-3	Pin 35 = Ch. B CTS

TABLE 59. DIG USART Jumper Settings

The following table gives the driver/receiver device types for both USART channels. Factory options allow for each channel to be independently populate with transceiver devices compatible with either the RS232 or RS422 standards. The table indicates which driver type will be populated in each part location depending on the option selected.

USART Channel	Line Standard	Populated Devices
0	RS232	U6, U7 = 7518x type
	RS422	U8, U11 = 26C3x type
1	RS232	U9, U10 = 7518x type
	RS422	U12, U13 = 26C3x type

TABLE 60. DIG Line Transceiver Device Types

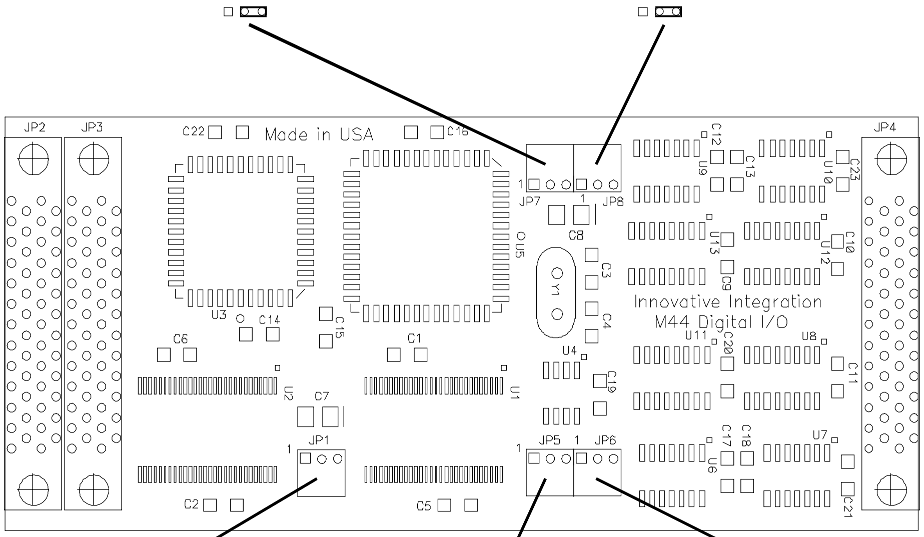
Factory Jumper Settings

JP7 - Channel B RTS/RX Clock Selector

1-2 I/O pin 39 = Receive Clock
2-3 I/O pin 39 = RTS

JP8 - Channel B CTS/TX Clock Selector

1-2 I/O pin 35 = Transmit Clock
2-3 I/O pin 35 = CTS



JP1 - Digital I/O Readback Clock

1-2 External Readback Clock
2-3 On-board Readback Clock

JP7 - Channel B RTS/RX Clock Selector

1-2 I/O pin 39 = Receive Clock
2-3 I/O pin 39 = RTS

JP6 - Channel A CTS/TX Clock Selector

1-2 I/O pin 43 = Transmit Clock
2-3 I/O pin 43 = CTS

CHAPTER 10 HSA Module

Module Introduction

The HSA module is designed for applications requiring high speed signal processing such as wireless communications (digital radio), RADAR, signal identification, transient capture and analysis and very high speed signal generation. The HSA module features a pair of high-speed A/D and D/A converters followed by a very large gate-count FPGA that is completely programmable for the user application. Other features such as the high-speed on-module, tunable timebase, high speed digital IO and memory subsystems give the module all the necessary features to construct a complex signal processing systems.

When the HSA is combined with a DSP card such as the M67, it is ideal for building applications such as testing and simulating 3G cellular radio or signal identification and tracking. Communications developers can use the FPGA to develop their signal processing techniques for a wide variety of applications, and then test them in the real world with the on-board analog IO.

The HSA module is delivered with VHDL code for the FPGA that is a framework for applications development. It shows some simple concepts like Omnibus interfacing, A/D and D/A control, DDS control and memory usage. It is by no means complete for any specific application, and it is expected that the user will develop the FPGA application and code. The framework is mainly a set of examples for the interfaces that may be used during development. This manual describes the use of the framework logic and the method for working with the framework to produce your application.

Application support for the HSA module is built around the Xilinx Foundation software. Xilinx and other vendors have many useful software cores that can be obtained (www.xilinx.com) to speed development.

Note: The HSA module is a complex, state-of-the-art system that requires advanced PGA programming. Users must be experienced in FPGA development and VHDL coding techniques to successfully employ the module. No beginners, please.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

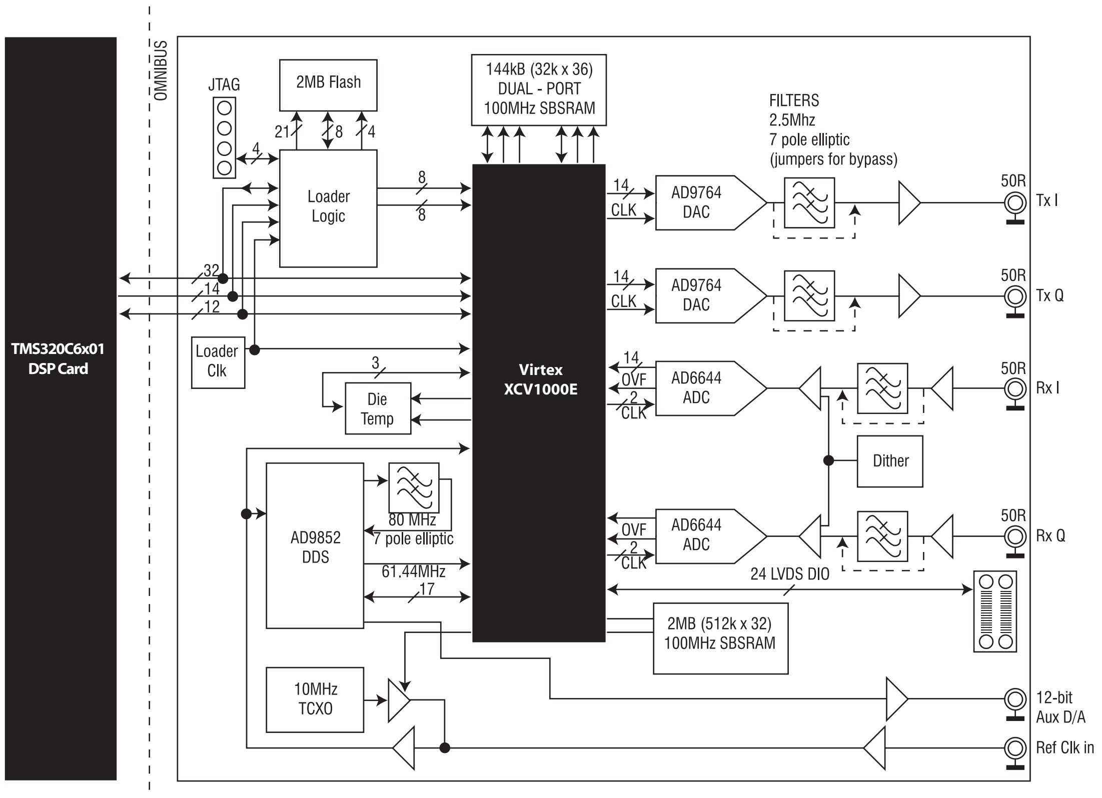


FIGURE 48. HSA Block Diagram

Bus Type:	Compatible with M6x & SBC6x cards. Contact Innovative Integration for support information on other platforms.	**D/A Converter:**	2 Analog Devices AD9764 converters.
Power Requirements:	6 Watts + app spec. logic usage; Forced air cooling required	**Resolution:**	14-bit.
Physicals:	Dual-wide Omnibus card; 4.0"X4.6"; Intrudes adjacent PCI slot when mounted on PCI card.	**Output Range:**	+/1 1V

A/D Converters	2 Analog Devices AD6644 converters.	**Settling Time:**	35 ns
Resolution:	14-bit.	**Dynamic Range:**	75 dB
Update Rate:	65 MHz max., 15 MHz min.	**THD:**	-72 dB
Analog Input Type:	Single ended to SMB connector.	**Offset Error:**	Trimmable on each channel - factory calibrated.
Analog Input Range:	+/- 1.65V	**Gain Error:**	Trimmable on each channel - factory calibrated.
Analog Input Impedance:	50 Ohms	**Interface to DSP:**	Memory mapped 32-bit result, programmable via Xilinx FPGA.
Input Filter Characteristics:	Low pass 7 pole; -3 dB point @ 12 MHz; jumperable.	**DDS - AD9852:**	300 MHz input with 40-bit resolution in frequency max. DDS output rate is up to 80 MHz.
SNR:	>65dB	**TCXO:**	5 ppm TCXO (10 MHz) onboard.
THD:	<0.01%	**Memories**	256K x 32 100 MHz SBSRAM
			32K x 36 100 MHz Dual Port
			8 MB Flash for FPGA logic.
Conversion Trigger Sources:	On board DDS or External clock; Optional dither on A/D for improved S/N.	**Temperature Sensor:**	MAX1617 temp. sensor for Virtex die temp. monitoring.
Interface to Host Card:	Memory Mapped 32-bit result, programmable via Xilinx FPGA.	**Digital I/O**	24 pairs LVDS I/O (Configured in logic)

HSA Module Software

The HSA development package includes support software and example applications, which illustrate how to code software to interact with the HSA hardware. This section of the documentation discusses the contents of the support libraries and explains the operation of the example application.

The included code assumes availability of and familiarity with the Zuma Toolset for the M6x DSP board. Execution, modification, and recompilation of the included source requires the Zuma Toolset. For information on how to use the Zuma Toolset with the M6x DSP card, please see the Zuma Toolset documentation.

Support is limited to code for Innovative's M6x DSP processor board. Support is not available for Innovative's other DSP board offerings, nor for the Chico family of data acquisition products.

Software Installation

The HSA example software is shipped as a self-extracting archive located on the Innovative Integration installation CD (d:\M6x\HSA_SW.exe). To install the software, run the executable file and select an installation directory when prompted. Use the default installation directory (the root of the C: drive) if you wish to make use of the provided Codewright project files included in the distribution. Click the Unzip button to install the file set.

During extraction, the files will be placed in a subdirectory CODE\M62\HSA of the selected installation directory.

Software File Contents

The following table gives a reference to the included source code organized by distribution filename.

File Name	Contents
BURN_FL.C	Logic configuration programming utility.
CONFIG.C	Logic configuration support code.
DDS.C	DDS control support code.
DDS.H	DDS control support definitions.
FLASH.C	Flash EEPROM support code.
FLASH.H	Flash EEPROM support definitions.
HSA.EXO	Example logic configuration image.
SBSRAM.C	SBSRAM support code.
SNAPSBS.C	A/D converter example application.
T_DAC.C	D/A converter example application.
T_DDS.C	DDS example application.
T_DIO.C	Digital I/O example application.
T_EXTCLK.C	External clock switching examples application.
T_SBSRAM.C	SBSRAM example application.

Software Functions

This section gives descriptions for each of the support functions included in the software distribution. The descriptions are organized according to the hardware feature they support.

The following figure gives a list of the support functions.

Functional Area	Function Name	Description
Logic Configuration (CONFIG.C)	HSA_config_virtex_from_flash()	Triggers logic configuration.
DDS Control (DDS.C)	HSA_init_DDS()	Initializes the DDS control register set.0
	HSA_set_DDS_freq()	Sets the DDS output frequency.
	HSA_run_DDS()	Deasserts reset to the DDS.
	HSA_reset_DDS()	Asserts reset to the DDS.
	HSA_select_DDS_clock()	Selects the DDS clock source.
	HSA_write_aux_DAC()	Writes data to the DDS auxiliary D/A converter.
Flash EEPROM Control (FLASH.C)	HSA_write_flash()	Write a byte value to the flash EEPROM.
	HSA_read_flash()	Read a byte value from the flash EEPROM.
	HSA_burn_flash()	Program a byte value in the flash EEPROM.
	HSA_flash_sector_erase()	Erase a single sector of the flash EEPROM.
	HSA_flash_erase()	Erase the entire contents of the flash EEPROM.
	HSA_write_flash_address()	Write an address value to the flash EEPROM address register.

Functional Area	Function Name	Description
	HSA_read_flash_address()	Read the current contents of the flash EEPROM address register.
	HSA_write_flash_data()	Write a data value to the flash EEPROM data register.
	HSA_read_flash_data()	Read the current contents of the flash EEPROM data register.
SBSRAM Access (SBSRAM.C)	HSA_write_sbsram_address()	Write an address value to the SBSRAM address register.
	HSA_write_sbsram_data()	Write data to the SBSRAM at the current address.
	HSA_read_sbsram_data()	Read data from the SBSRAM at the current address.

Logic Configuration

Prior to accessing memory mapped registers implemented within the Virtex device, the device must be configured from the contents of the flash EEPROM memory. Calling the **HSA_config_virtex_from_flash()** function accomplishes this configuration.

If the flash EEPROM contents are modified (for example, if the logic design is updated), the Virtex device must be reconfigured using an additional call to **HSA_config_virtex_from_flash()**.

DDS Control

The DDS support functions allow host software to initialize the DDS timebase, program its output frequency, select from several clock sources, and output signals on the DDS auxiliary output channel.

The DDS initialization is accomplished using the **HSA_reset_DDS()**, **HSA_run_DDS()**, and **HSA_init_DDS()** functions. The first two functions are used to physically reset the DDS device as a first step to initialization to ensure that the DDS register set assumes a known startup state. The **HSA_init_DDS()** function activates the DDS' internal PLL with a multiplication factor of x20 (resulting in a 200 MHz internal clock if the standard 10 MHz onboard oscillator is used) and enables internal clock update mode. This initialization is suitable for most applications, but some may require other configurations: see the source code for the function and the 9852 DDS data sheet for details on the control register contents.

HSA_select_DDS_clock() is used to select the clock source which the DDS will use as its clock reference for timebase generation. Calling the function with an argument of 0 will select the onboard 10 MHz TCXO oscillator, while calling the function with an argument of 1 will select the external clock connection as the signal source.

The **HSA_set_DDS_freq()** function is used to program the DDS output frequency. The function takes two arguments: the first is the desired output frequency (in MHz), the second is the internal source clock frequency (also in MHz). The source clock frequency is the input clock reference multiplied by the DDS PLL multiplier (if used). By default, the initialization software selects the 10 MHz onboard oscillator and a PLL factor of x20, so the source clock frequency would be set to 200 MHz. Updates of the DDS output frequency occur on a software programmed interval dictated by a DDS internal timebase (see the 9852 DDS data sheet for more information on the internal update feature).

The DDS implements a second D/A channel which may be software controlled using the **HSA_write_aux_DAC()** function, which takes a single argument which it writes to the D/A data register. Input data coding for the converter is 12 bit two's complement. The converter's output is updated using the internal update clock (see above).

Flash EEPROM Control

The flash EEPROM memory provides nonvolatile storage of configuration data for the Virtex logic device. This data is used to configure the logic device on powerup following a configuration request (see above). Several functions are included in the library to allow the host processor to write and read data in the EEPROM, allowing the processor to update the configuration data used by the Virtex device.

Data may be reads from the EEPROM using the **HSA_read_flash()** function. The function takes a single argument, the byte address of the data to be read. The function returns the value of the byte at the required address. Returned data is masked to the least significant byte by the function (i.e. no additional masking is required by the application program calling the function).

Data may be written to the flash device using the **HSA_burn_flash()** function. Two arguments are used by the function: the first is the byte address of the location to be written, and the second is the value to be written to the device. The function executes a set of command writes to the EEPROM to trigger an internally managed write mechanism, then pauses while the write completes. The location to be written must have first been erased by using either the **HSA_flash_sector_erase()** or **HSA_flash_erase()** functions (see below) prior to starting the write.

An individual sector may be erased using the **HSA_flash_sector_erase()** function. The function takes a single argument, the sector number to erase. Following a string of command writes to trigger the erasure, the function pauses to allow the operation to complete.

The entire flash EEPROM device may be erased using the **HSA_flash_sector_erase()** function. The function takes no arguments. Following a string of command writes to trigger the erasure, the function pauses to allow the operation to complete.

Numerous additional EEPROM access functions are included in the software distribution: these are subfunctions used by the above described functions, and generally do not need to be called directly by user software.

SBSRAM Access

Onboard SBSRAM may be accessed by host software using the SBSRAM access functions. The SBSRAM is accessed by the logic through the use of an address latch which holds the address in SBSRAM. Software first writes the address to access, then executes either a read or write operation to the SBSRAM.

The **HSA_write_sbsram_address()** function is used to set the address in SBSRAM to be accessed. The function takes a single argument, the address in SBSRAM. This value is written to the SBSRAM address register and the function immediately returns.

Once the address is set up, data accesses are performed using the **HSA_write_sbsram_data()** function (for writes) and the **HSA_read_sbsram_data()** function (for reads). **HSA_write_sbsram_data()** takes a single argument, the data value to be written. The write operation is performed and the function returns. The **HSA_read_sbsram_data()** function takes no arguments, performs the read operation, and returns the data read from SBSRAM.

Example Programs for the HSA Module

In addition to the support functions, the software distribution contains several example programs showing how software is written to use the support functions to interact with the HSA hardware configured with the framework logic.

In order to execute the example programs, the HSA must be installed on an M6x host DSP card which is in turn installed in a host PC running the Zuma Toolset. The example programs make use of stdio function calls and so must be run from within the UniTerminal environment.

<u>BURN_FL.C</u>

This example shows how to update the flash EEPROM with new data read from a Xilinx toolset-generated .EXO file residing on the host PC. The host UniTerminal file I/O functions are used to read data from .EXO file, whereupon it is written to the flash EEPROM memory via the support functions.

See the Developing FPGA Firmware section above or information on creating the .EXO file using the Xilinx tools and updating the HSA board EEPROM memory.

<u>SNAPSBS.C</u>

Shows how to acquire A/D data from the converters to the SBSRAM memory on the HSA module, based on timing provided by the DDS clock source. The program also presents a graphical plot of the acquired data via the UniTerminal plotting feature.

<u>T_DAC.C</u>

Enables the D/A converters to generate a hardware-generated ramp output.

<u>T_DDS.C</u>

Shows how to initialize the DDS clock source to produce timebase information. Also shows use of the DDS auxiliary D/A output.

<u>T_DIO.C</u>

Outputs test pattern data via the digital I/O port.

<u>T_EXTCLK.C</u>

Shows how to select the external clocking features of the HSA hardware.

T_SBSRAM.C

Exercises the SBSRAM interface using the support functions.

HSA Module Hardware

FPGA

The HSA module includes a state-of-the-art high-density FPGA that is used for signal processing, peripheral control and as an interface to the host DSP card. This device is a Xilinx Virtex 1000E device (other devices may be ordered, contact Innovative Integration sales department) which is a one million gate FPGA. This FPGA is extremely fast, flexible and has support for many of the functions needed to create a complex signal acquisition and processing system.

Support for programming the FPGA is provided by the Xilinx Foundation toolset. Starting with the basic framework logic that is delivered with the module, the user can develop the application logic that is run in the FPGA. This framework logic is delivered as VHDL code

Analog Interface

The HSA module features a pair of 65 MSPS, 14-bit A/D channels for digitizing input signals. The input signal conditioning provides either a bandpass or lowpass filter for the input signals. The A/Ds have a direct data path to the FPGA to support high sample rates directly to the FPGA. Direct control of all the A/D control signals allows the user application to trigger the A/Ds in a variety of ways. These include the timebase, DDS, or externally either in quadrature or synchronization as required.

A/D

The HSA module has a pair of AD6644 A/Ds that are capable of 65 MSPS, 14-bit conversions each. The minimum sample rate is 15 MHz on the AD6644. For applications requiring sample rates below 15 Mhz, it is suggested that the application average multiple samples or simply decimate higher rate data.

The A/Ds on the HSA module each have a direct data path to the FPGA and are each directly controlled by the FPGA. This allows the user application to be designed so that the high-speed data is directly available to the FPGA and the control of the A/Ds may be designed as necessary for the application to trigger the samples as needed.

The A/D converters interface directly to the logic on independent data paths. Each A/D has a 256 sample FIFO in the logic. The data format from the A/D converters is 2's complement with the overflow bit appended on bit 15 of each word. Bit 14 of the word is always '0'.

A/D data = 'OVR ' & '0' & data [13..0]

In the example FPGA framework code, each A/D has a 256 sample FIFO inside the FPGA. Triggering for the A/Ds is currently hardwired to the host DDS (limits test to 25 MHz sample rate). This example demonstrates how to get the data from the A/Ds and put it into a simple FIFO. The A/D FIFO levels are monitored in the FIFO status register and may also be used to trigger module interrupts. In addition, the sample FPGA code shows the PECL drivers required for the A/D conversion clock signals. These are required and must not be deleted from the code.

This A/D achieves an approximate SNR of 60-65 dB, 70-75 dB noise floor, 20-30 count p-p in time domain with grounded input.

Input Amplifiers and Filtering

The input signal path includes buffering, an analog filter and conversion to a differential signal pair before entering the A/D. The following figures give the schematics for the front-end circuitry.

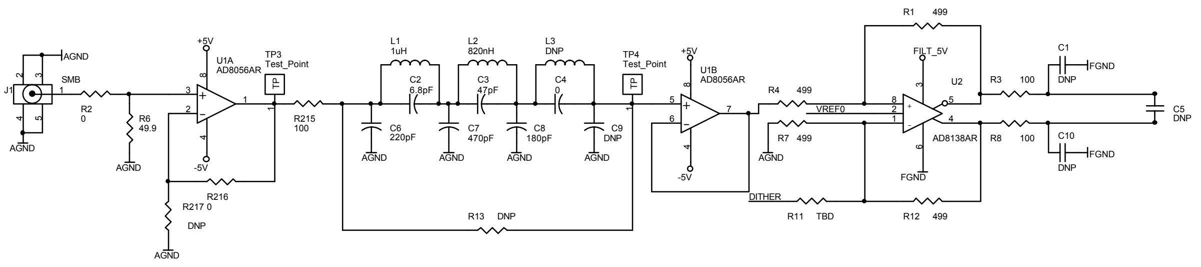


FIGURE 49. **HSA A/D Channel 0 Input Circuitry**

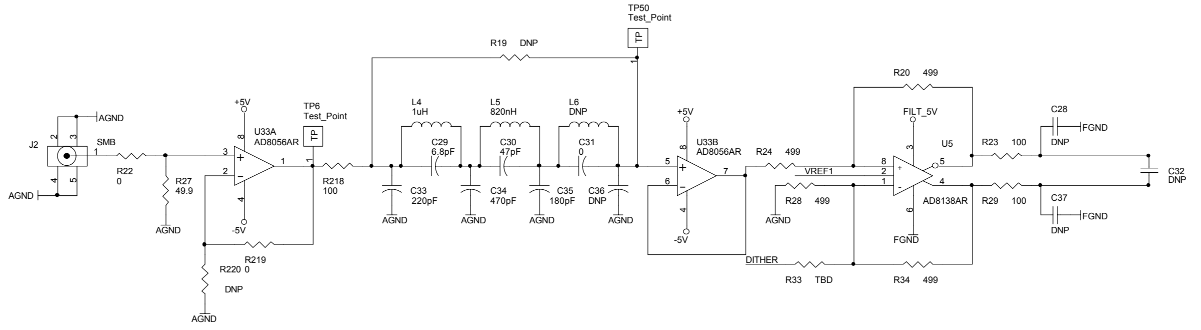


FIGURE 50. **HSA A/D Channel 1 Input Circuitry**

Each input is single-ended input with a 50-ohm termination impedance. For the best performance, it is important to match the signal source impedance and cabling so that signal reflections and distortion are minimized. The input connector is a 50-ohm SMB connector for coaxial cable connection. The default input impedance is 50-ohms. Standard input range is +/-1.1V (2.2Vp-p). This can be modified by changing the pair R2/R6 to allow higher input signals at the expense of input impedance matching. Lower level input signals can be accommodated by changing the gain on the buffer amplifier stage. The default gain is one. Other gains may be achieved by changing the buffer amp resistor pair.

The input filter is a 12 MHz lowpass filter by default, which allows the HSA to reject out-of-band noise before digitizing. This filter may be bypassed and disabled if required by installing jumpers R13 and R19 for inputs A and B respectively. If the filters are bypassed, the input channel bandwidth is very high and can be used for undersampling applications for direct digital downconversion. Custom filters may also be ordered: contact the sales department at Innovative Integration.

The final amplifier stage converts the single-ended input to a differential input for the A/D.

D/A

The HSA module has dual 100 MSPS, 14-bit D/A converters. These D/A channels are useful as transmit channels or waveform generators. The AD9764 D/A is specifically designed for transmit applications where spectral purity is paramount. The output signal conditioning provides a lowpass filter to minimize out of band noise. The A/Ds have a direct data path to the FPGA to support high sample rates directly to the FPGA. Direct control of all the A/D control signals allows the user application to trigger the A/Ds in a variety of ways. These include the timebase, DDS, or externally either in quadrature or synchronization as required.

Output Amplifiers and Filters

The D/A output includes a 12 MHz lowpass filter. The filter may be bypassed if DC output is required. The standard output range is +/- 1V. The output connector is a 50-ohm SMB connector for coaxial cable. It is important the cable and terminal impedance match this output impedance for best performance. The following figures give the output circuitry schematics.

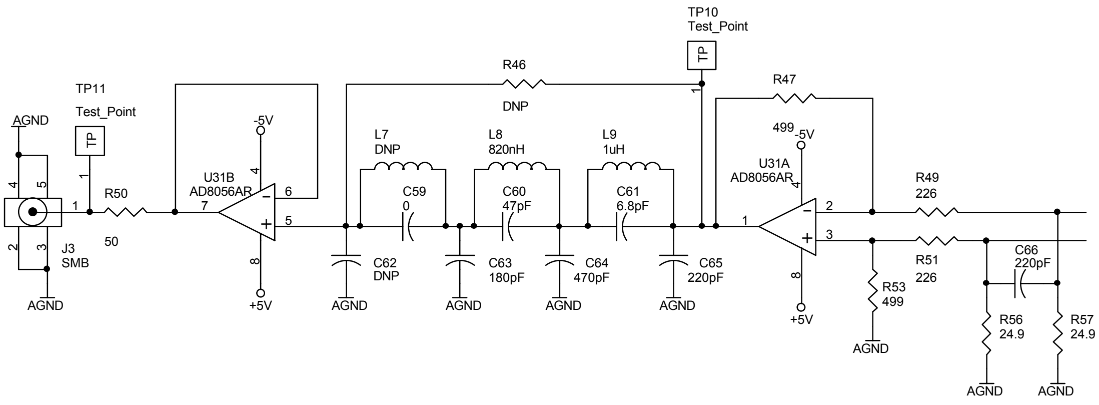


FIGURE 51. **HSA D/A Channel 0 Output Circuitry**

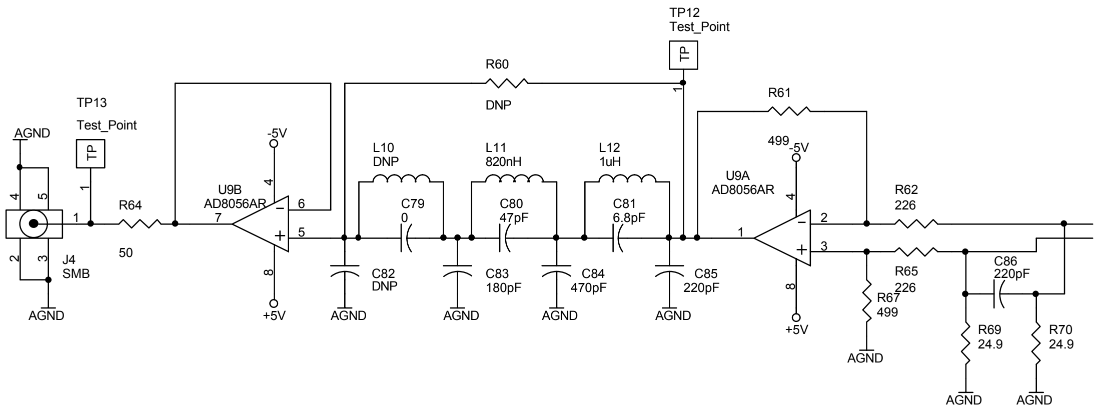


FIGURE 52. **HSA D/A Channel 1 Output Circuitry**

D/A Interface

The framework example code for the FPGA shows the D/As directly fed by a 256-sample FIFO. The DAC FIFO levels are monitored in the FIFO status register and may also be used to trigger module interrupts.

In the example, the triggering for the DACs is currently hardwired to the host DDS (limits test to 25 MHz sample rate). This may be modified with relative ease to use the on-module DDS for higher rates or to any of the external signals depending upon the application.

DDS

The HSA module has an AD9852 Direct Digitally Synthesizer (DDS) that is useful as a timebase and waveform generator. The DDS can generate any frequency from 0 to 100 MHz with 1 uHz resolution so it is ideal for use as a local oscillator for demodulation in fixed or frequency agile systems.

DDS Timebase

The DDS timebase is normally the 10 MHz TCXO (Temperature Compensated crystal Oscillator). This TCXO has a stability of 5 ppm and may be calibrated for absolute accuracy of the frequency. This TCXO is routed into the FPGA, which may then route the clock to DDS input. The DDS has an internal frequency multiplier that makes a much higher frequency used by the DDS.

DDS Control and Programming

The interface to the device consists of an address and data register for communicating with the DDS internal registers, a DDS control register for several DDS mode pins, and a clock steering register for selecting the DDS clock source.

The DDS interface registers are mapped as follows:

Bits	Function
5..0	DDS address 5..0

TABLE 61. HSA DDS Interface Register

The data path to the DDS is byte wide and corresponds to bits 7..0.

The DDS control bits are mapped as follows

Bit Number:	0	1	2
Bit Field:	DDSRST	DDSSHP	DDSFSK

FIGURE 53. HSA DDS Control Register

Bit Field Name	Function
DDSRST	DDS Reset (default is ‘1’ in reset)
DDSSHP	DDS Shape (default is ‘1’)
DDSFSK	DDS FSK (default is ‘1’)

TABLE 62. HSA DDS Control Register Definition

The DDS reference clock may be selected from the on-board TCXO or the externally supplied clock. Writing a 0 selects the corresponding clock source, writing a 1 deselects that clock source. Only one clock source should be enabled at a time (i.e. write the value 2 to the register to select the TCXO source; write a 1 to select the external reference clock).

Bit Number:	0	1
Bit Field:	TCXO	EXTERNAL

FIGURE 54. HSA DDS Reference Clock Register

Bit Field Name	Function
TCXO	TCXO Reference Clock (default = on)
EXTERNAL	External Reference Clock (default = off)

TABLE 63. **HSA DDS Reference Clock Register Definition**

The DDS external reference clock input connector is J5. This 50 ohm impedance SMB connector is terminated to ground with a 50 ohm resistor. The external clock input signal can be TTL compatible or may also be a sine wave with a swing of +/-3V maximum referenced to ground.

Two connectors are provided to allow the user to connect to the DDS primary and secondary channel outputs. 50 ohm SMB connector J6 provides a bipolar sine wave output of approximately +/-500mVat the programmed DDS output frequency (i.e. the same frequency delivered to the analog converters and logic). 50 ohm SMB connector J7 provides access to the DDS secondary channel (see the 9852 datasheet for details on the uses of the secondary channel).

The DDS primary and secondary output channel signal chains contain independent passive filters which may be disabled by installing shorting jumpers on headers JP1 and JP2, respectively. For normal DDS operation the jumpers should not be installed.

Memory

The HSA module has two memory sub-sections that allow the user application to have a large pool of high-speed memory available for computation, buffering, or scratch memory. These memories have separate data path and control signals so that the user can use the memories as necessary in the FPGA processes.

SBSRAM Memory

The HSA module has 256K x 32 SBSRAM (synchronous burst SRAM) memory comprised of one Micron Technology (MT58L256L32FS-10) chip. This device has a flow-through architecture. The SBSRAM supports up to 100 MHz (400 Mbyte/sec) data rates during burst accesses. This memory may be used by the FPGA in any process as required by the application and has completely independently controls. Since this SBSRAM provides the highest data rates only in burst mode, it is suggested that the user arrange data to take advantage of this architecture. The sample framework code provided for the FPGA has a simple SBSRAM interface the merely demonstrates that the RAM is connected and working. It consists of two registers: address and data. To use, write an address to the SBSRAM address register, then read or write from that address. More complex control is left to the application as required.

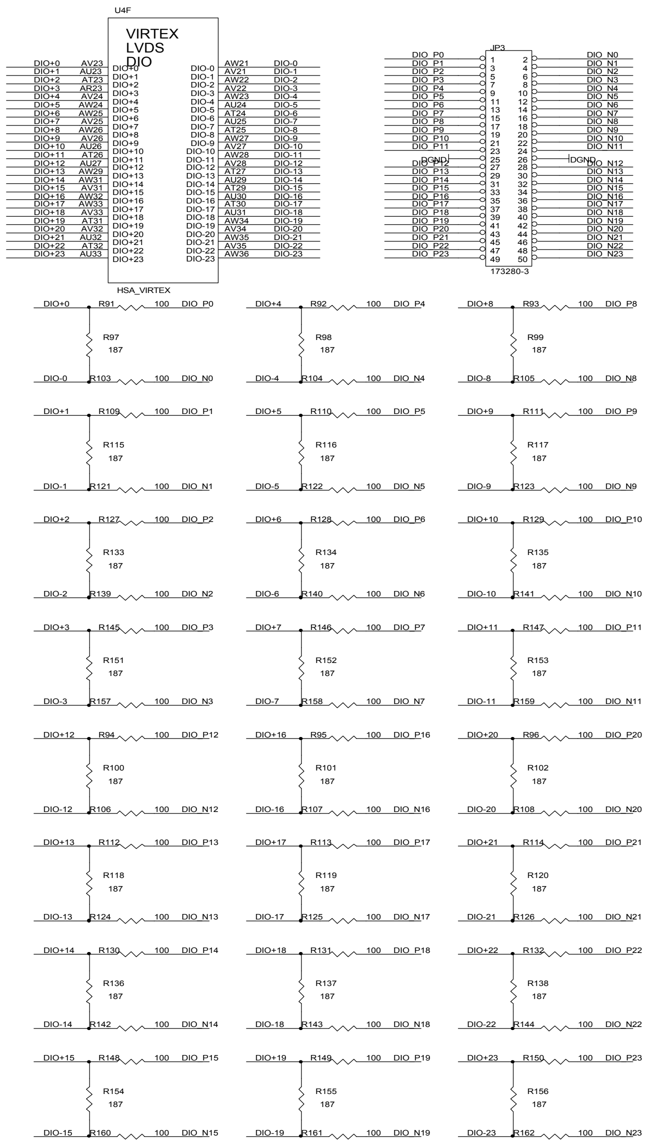
Dual Port SBSRAM Memory

The second external memory pool on the HSA module is a dual-port SBSRAM, Cypress Semi CY7C09579. This memory is 32Kx32 with two independent interfaces into the memory. This memory is most useful as a means to pass a data set between processes in the FPGA, but can also be used as general-purpose memory. Each side of the dual port memory has complete access to the memory so that simultaneous accesses are possible. This memory is capable of up to 100 MHz (400 Mbyte/sec) data rates during burst accesses. Since this Dual Port SBSRAM provides the highest data rates only in burst mode, it is suggested that the user arrange data to take advantage of this architecture.

The sample framework code provided for the FPGA has a simple interface that merely demonstrates the RAM is connected and working. More complex control is left to the application as required. In the example code, the two sides of the memory are referred to as left and right sides. It consists of four registers : left address, left data, right address, and right data. To use the register, write an address to the SBSRAM address register, then read or write from that address. Note that the concurrent accesses to the same address are not possible under this simple interface so no collision will occur. More complex interface schemes may require the application to respect the dual port concurrent memory access restrictions.

Digital I/O Interface

The digital I/O interface consists of 24-bits of digital I/O, configured as 12-pairs of LVDS signals. The digital IO pins have resistor networks configuring them as LVDS outputs as shipped. These can be reconfigured as inputs by reprogramming the logic and removing the series resistor on each leg. The terminating resistor should be 200-ohms. The logic may also be reprogrammed as LVTTL for convenience, but then all resistors on the signal pins should be removed. See schematic below for details.



In the application framework, the digital IO is controlled by a set of registers for the data and direction control.

Direction of the signals is programmable using the digital I/O configuration register, but as a practical measure may not be changed on the existing card without changing resistors on the pins used in LVDS mode.

The bits in the digital I/O register are assigned as shown here.

Bits	Function
23..0	Digital I/O address 23..0

TABLE 64. **HSA Digital I/O Register**

The digital I/O configuration register is as shown here. Direction is controlled on a byte-by-byte basis. Setting each bit in the register to a value of 0 selects input mode for the corresponding port byte, while setting the register bit value to 1 selects output mode.

Bit Number:	0	1	2
Bit Field:	DIO-0	DIO-1	DIO-2

FIGURE 55. **HSA Digital I/O Configuration Register**

Bit Field Name	Function
DIO-0	DIO byte 0
DIO-1	DIO byte 1
DIO-2	DIO byte 2

TABLE 65. **HSA Digital I/O Configuration Register Definition**

Readback of latched data is only possible on bytes programmed for output mode.

Temperature Monitor Interface

The temperature monitor can report the die temperature of the Virtex logic for monitoring. The high clock rates and density of the Virtex device make it common for the logic design to consume 3-6 W. Dissipating this amount of power will require attention be paid to proper heat-sinking and cooling air-flow.

The temperature monitor is accurate within a few degrees C. The interface gives a simple register based control to Maxim Semi MAX1617. This is a bit-banged serial interface, so write the data then pulse the clock low to high then high to low. See the MAX1617 data sheet for the required protocol.

Auxiliary Clock Oscillator/Input

The HSA module provides for connecting an independent clock source to the Virtex device via the AUX_CLK pins. The clock source may be either an onboard oscillator installed at location U21, or an external LVDS compatible signal.

If an onboard oscillator is used it must be a 3.3V compatible device package in an SG615 case installed at location U21. In addition, R174 MUST be depopulated or contention with the LVDS receiver device at U22 will result.

If an external signal is to be used it must be LVDS compatible, with the positive side connected to pin 49 of the OMNIBUS host's I/O connector for slot 0, and the negative side of the signal connected to pin 50. Oscillator U21 must be depopulated in this case, and a zero ohm resistor installed at position R174.

Heatsinking and Power Dissipation

The FPGA on the HSA module can dissipate over 9 Watts of power when the logic is fully utilized and is running at a high clock rate. In addition, the A/Ds, D/As, DDS, and power supplies add over 4 Watts to this power dissipation when running at high clock rates. Therefore, careful attention should be paid to the heat sinking and power dissipation of the module.

The FPGA design should be carefully examined to determine the expected heat dissipation. The framework code is very low usage and only dissipates about 1W. No heatsinking is necessary for this framework logic. More advanced designs however, will require the user to carefully assess the power consumption expected from the FPGA and be sure than adequate heat sinking is made on the module. Normally a local cooling fan is required on the FPGA for advanced designs.

As shipped, the A/Ds have a passive heat sink on them. Proper operation of the module requires that forced air be flowing past the module. Depending on the system chassis design, this may require an additional fan be mounted close to the HSA module. Many industrial chassis have sufficient air flow to cool the A/Ds properly.

Caution: Failure to properly cool the FPGA will result in damage to the HSA module.

Reading the Logic Version

The module version number is programmed into the non-volatile logic and may be read back on this register. Current logic version is "F". This may only be reprogrammed using the Xilinx download cable.

Accessing the ID ROM

The module IDROM resides at the standard IOMOD3 0-1 addresses and is identical to the AD16 module interface.

HSA Module Physicals

The HSA module is a dual-slot Omnibus module. The module mechanicals, components, and connections are show in the following figures below:

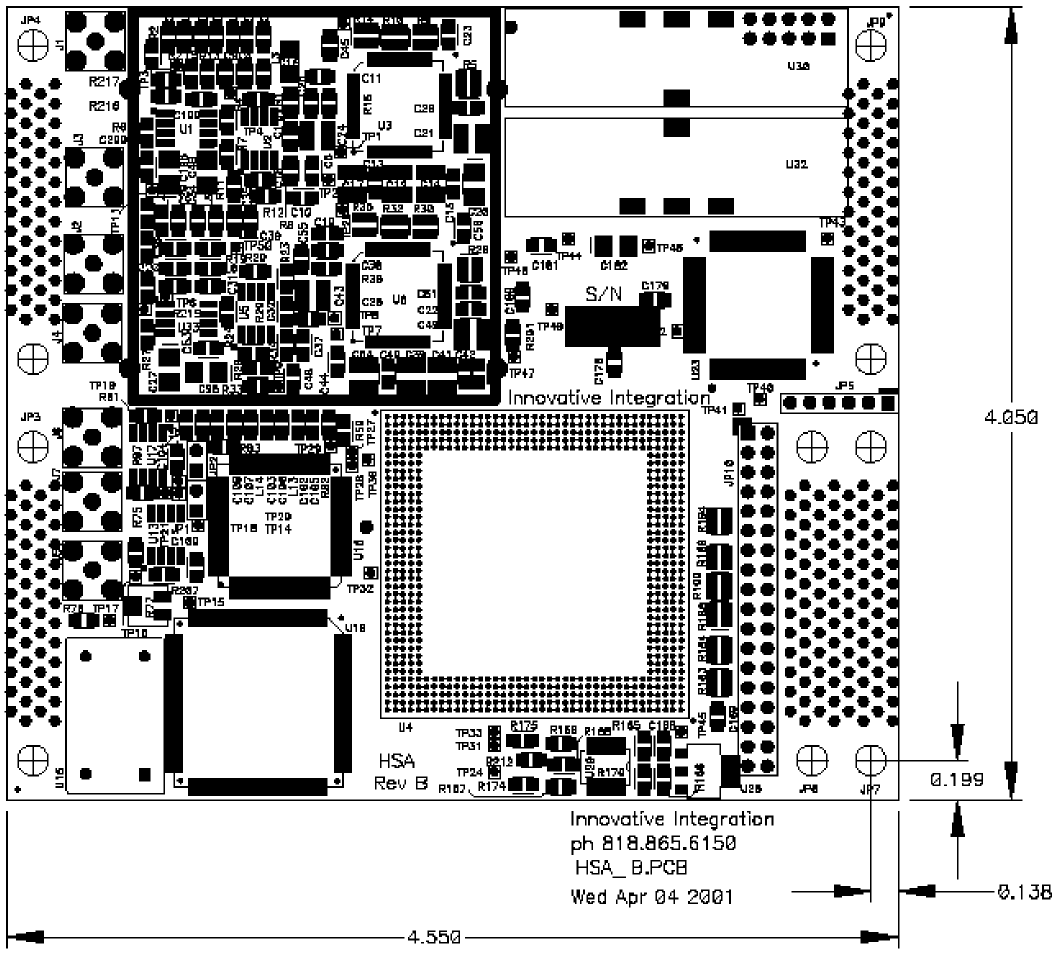


FIGURE 56. HSA Module Dimensions

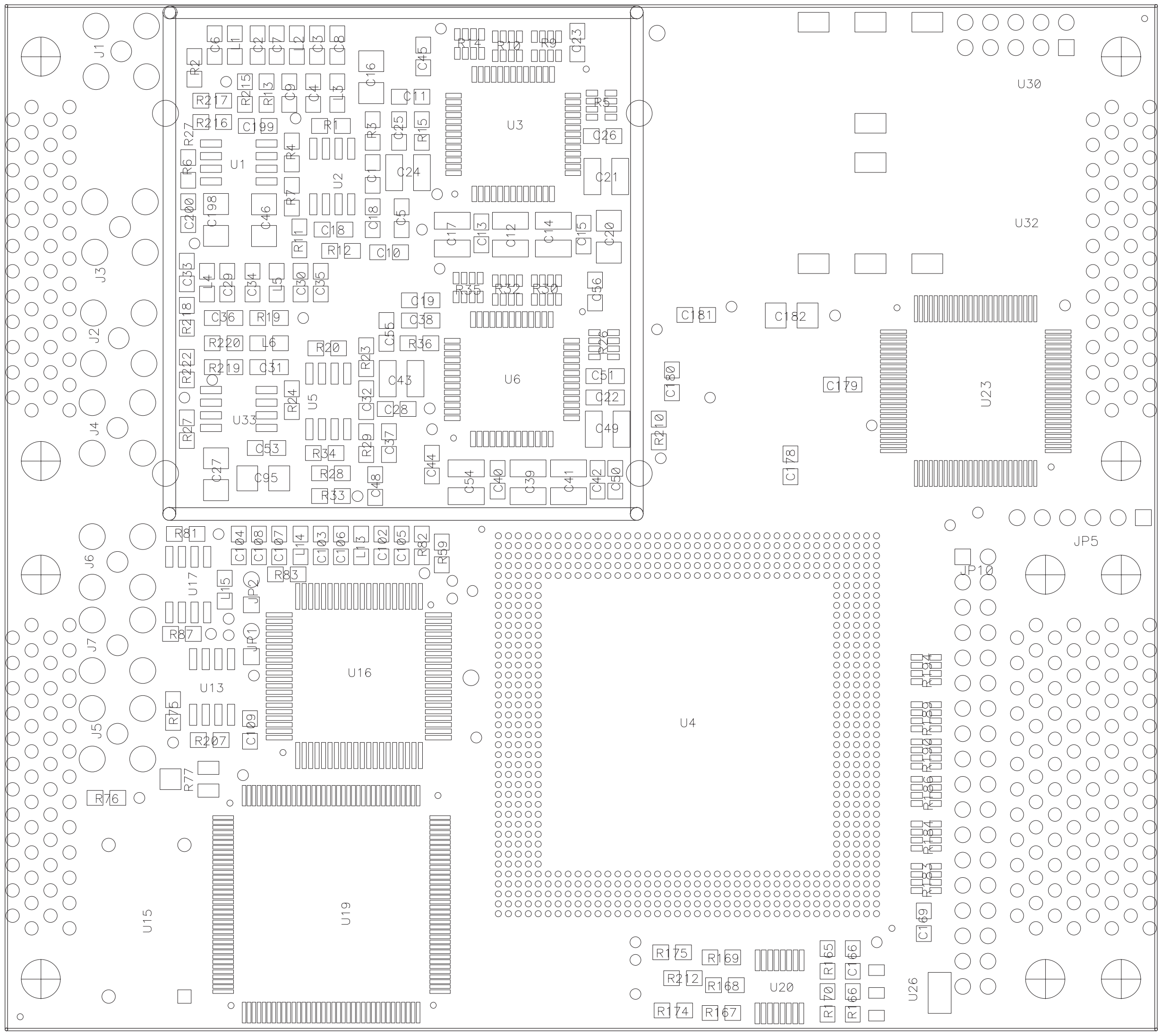


FIGURE 57. **HSA Module Top View**

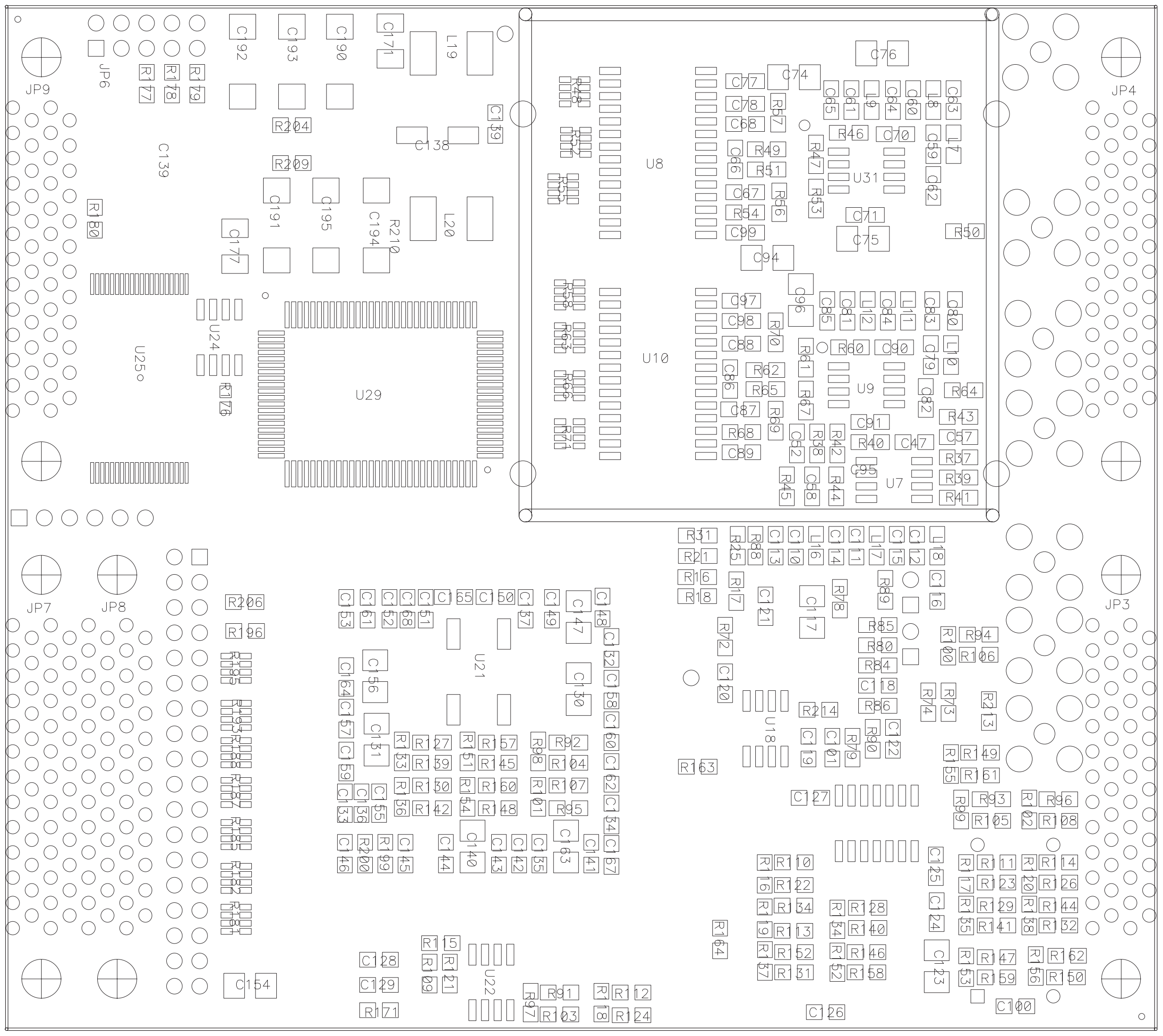


FIGURE 58. **HSA Module Bottom View**

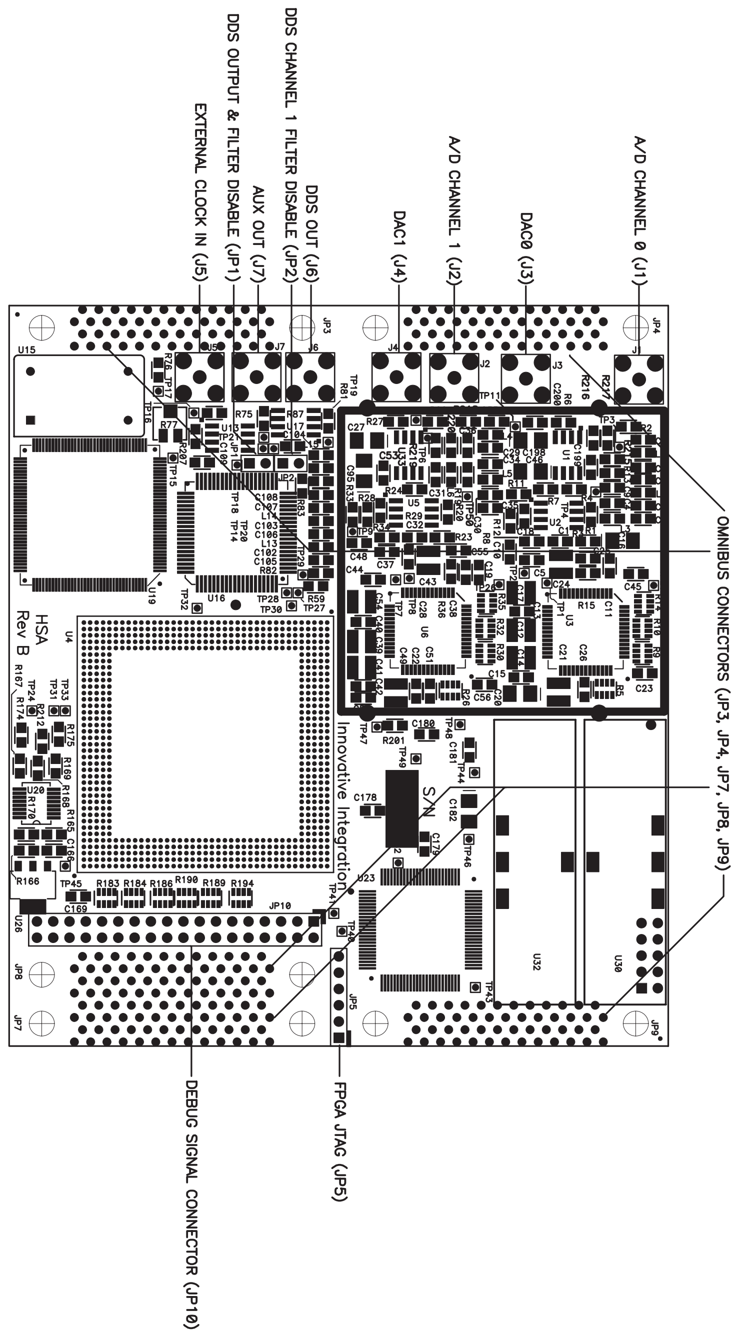


FIGURE 59. **HSA Module Connector Layout**

Note that when the HSA module is mounted on an M67 board, it will intrude into a second PCI slot. Right angle SMB cables are recommended so that a third PCI slot is not obstructed.

FPGA Loading Modes

Boot Modes for logic Loading

The logic may be loaded either from the on-board FLASH memory or written directly by the host into the SelectMap interface. The bootloading process is controlled by the non-volatile logic on the HSA module. The non-volatile logic is always present on the module so that bootloading functions even when the FLASH memory is erased or contains faulty logic. The default is the download from FLASH. All pins are pulled high during the booting process by a weak pull-up.

Control Register for Logic Loading

This register controls the logic loading mode, the init and program pins on the Virtex FPGA.

Bit	Function
0	Program bit to flash, default is false (pin is active low) 1 = program
1	Init bit: assert (‘1’) then release to reload logic.
2	Not Used
3	Mode: 0 = load from flash, 1 = load from host
4	Not Used

TABLE 66. **HSA Control Register for Logic Loading**

Programming the FLASH Memory

Software routines are provided to allow the user to burn logic files into the boot memory. These routines allow you to burn the FLASH memory on HSA module from an EXORMacs format that is generated as described in the following section on a M6x DSP card.

To program the flash memory, the host must set the mode to '1' and the program bit to '1', then it can program the flash using the flash address registers and data register. Complete details for the flash programming routine is described in the AMD data sheet for the AM29LV116B. The flash address registers hold the 3 bytes needed to describe the 21-bit flash address. Each byte is located at a different address, but all use bit 7..0.

Accessing the Logic Using SelectMap

The SelectMap mode of configuring the Xilinx logic is a useful means to load the HSA logic as part of the DSP application software or during the debug of the FPGA logic. Dynamic logic configuration is possible using full or partial reconfiguration using SelectMap accesses.

In SelectMap mode (mode = 1), the host can read and write directly into the Virtex to configure the logic or partially configure the device while in operation. In this mode, host reads and writes to the SelectMap data register directly talk to the Virtex SelectMap. Refer to Xilinx documentation for complete details on configuration in this mode.

FPGA Firmware

Overview

The HSA module has a Xilinx Virtex FPGA. Developing logic code for this FPGA is done using the Xilinx Foundation Toolset and the framework logic provided. The framework logic is delivered in VHDL language, along with the control files for the Xilinx Foundation software to allow the user to modify and recompile the logic. This framework logic gives minimal interface and control for the various peripherals and is provided only as a means to get started coding with FPGA.

Using FPGA Framework

The FPGA Framework logic maps all of the HSA peripheral devices to allow access to the various control registers and data required to use the device. This memory mapping is done using four IOMOD regions on the Omnibus interface. Each Omnibus IOMOD is further decoded as shown in the following memory map. IOMOD 3 is typically reserved in Omnibus designs for the ID ROM and other version information so that the supporting software has a consist interface with these functions.

IOMOD	Address (Hex)	Device	R/W
4	0	A/D 0	R
	1	A/D 1	R
	2	DAC 0	W
	3	DAC 1	W
5	0	SBSRAM Address	W
	1	SBSRAM Data	R/W
	2	DPRAM Left Address	W
	3	DPRAM Right Address	W
	4	DPRAM Left Data (32-bit)	R/W
	5	DPRAM Right Data (32-bit)	R/W
6	0	DIO	R/W
	1	DIO Direction Configuration	W
	2	Temperature Monitor Data	R/W
	3	Temperature Monitor Clock	W
	4	Interrupt Configuration	W
	5	Control Register	W
	6	FIFO Status Register	R
	7	DDS Control Register	W
	8	DDS Clock Control Register	W
	800	DDS	R/W

7	0	IDROM Data	R/W
	1	IDROM Clock	W
	2	Module Control Register	R/W
	3	Version Readback	R
	4	FLASH Data	R/W
	5	FLASH Address byte 0 (LSB)	R/W
	6	FLASH Address byte 1	R/W
	7	FLASH Address byte 2	R/W
	8	FLASH Address byte 3	R/W

TABLE 67. HSA Memory Map

Module Interrupt Configuration

The interrupt from the module on MOD INT 0 is controlled by this register. Each FIFO level in the FIFO status register may be enabled as an interrupt in a one-to-one correspondence with the bits in the FIFO status register. Set the bit to '1' to enable the interrupt source. All interrupt sources are or'd to produce the module interrupt.

Bit 31 enables the interrupt and must be set to '1' to enable the interrupt. This interrupt enable bit is normally set after all other configuration to the module is complete and the DSP is ready to receive data.

The module may interrupt the host on MOD INT 0 when the interrupt is enabled and the interrupt configuration for FIFOs is not zero. These are both in the interrupt configuration register.

The software should read the FIFO status register and to determine which FIFOs require service, if multiple FIFO interrupts are enabled.

Bit	Interrupt Enabled
0	DAC0 FIFO not empty
1	DAC0 FIFO not half full
2	DAC0 FIFO full
3	DAC1 FIFO not empty
4	DAC1 FIFO not half full
5	DAC1 FIFO full
6	ADC0 FIFO not empty
7	ADC0 FIFO half full
8	ADC0 FIFO full
9	ADC1 FIFO not empty
10	ADC1 FIFO half full
11	ADC1 FIFO full
31	Enable Interrupt

TABLE 68. HSA Interrupt Configuration Register

Module Control Register

The module control register contains reset bits for the FIFOs. These may be used to clear the DAC and A/D FIFOs at any time.

Bit	Function
0	Clear both DAC FIFOs (0 = clear)
1	Clear both A/D FIFOs (0 = clear)
2	A/D FIFO load enable (0 = disabled, 1 = enabled)
3	D/A FIFO read enable (0 = disabled, 1 = enabled)

TABLE 69. **HSA Module Control Register**

FIFO Status Register

The FIFO status register allows the current FIFO levels to be read by the host. These are not latched so they can change during reading. Each FIFO has its not empty, half and full flags available in this register.

Bit	Function
0	DAC0 FIFO not empty
1	DAC0 FIFO half
2	DAC0 FIFO full
3	DAC1 FIFO not empty
4	DAC1 FIFO half
5	DAC1 FIFO full
6	ADC0 FIFO not empty
7	ADC0 FIFO half
8	ADC0 FIFO full
9	ADC1 FIFO not empty
10	ADC1 FIFO half
11	ADC1 FIFO full

TABLE 70. **HSA FIFO Register**

Developing FPGA Firmware

Generating Virtex Configuration Files and Programming the Configuration Flash Memory

Virtex designs for the HSA module are compiled under Xilinx Foundation version 2.1i service pack 6. The example/debug logic design is a Project Manager project loadable by opening Project Manager and performing a Project | Open on the HSA.PDF file in the logic root directory. The design directory path originally used to house this project hierarchy was **`C:\PROJECTS\HSA\LOGIC\HSA`** (i.e. the .PDF file appears in **`C:\PROJECTS\HSA\LOGIC`**, one directory above the source .VHD files which appear in the HSA subdirectory). It is recommended that the customer's installation duplicate this directory configuration in order to avoid problems with the Xilinx tools.

Synthesis is performed under the Project Manager by clicking on the Synthesis flow button, once the project is loaded. Once synthesis completes, the user must open the Xilinx Design Manager to complete place and route on the design. This is due to the lack of device availability for the target device (standard is Virtex XCV1000E-FG680) under the 2.1i Project Manager. Start the Device Manager by selecting its entry from the Start Menu | Programs | Xilinx Foundation Series 2.1i | Accessories menu on the desktop. Open the current version of the design using File | Open Project, and implement the design by selecting Design | Implement. The design must be implemented using the included constraints file (HSA.UCF) in order to duplicate pin definitions correctly.

Finally, an EXORMacs format text file must be generated from the output .BIT file produced by the place and route process. This is done by opening the PROM File Formatter utility from within the Design Manager. Open the included PROM description file (HSA.PDR) by selecting the File | Open command. Select File | Save to convert the .BIT file to the .EXO EXORMacs format. The .EXO file is generated in the source directory.

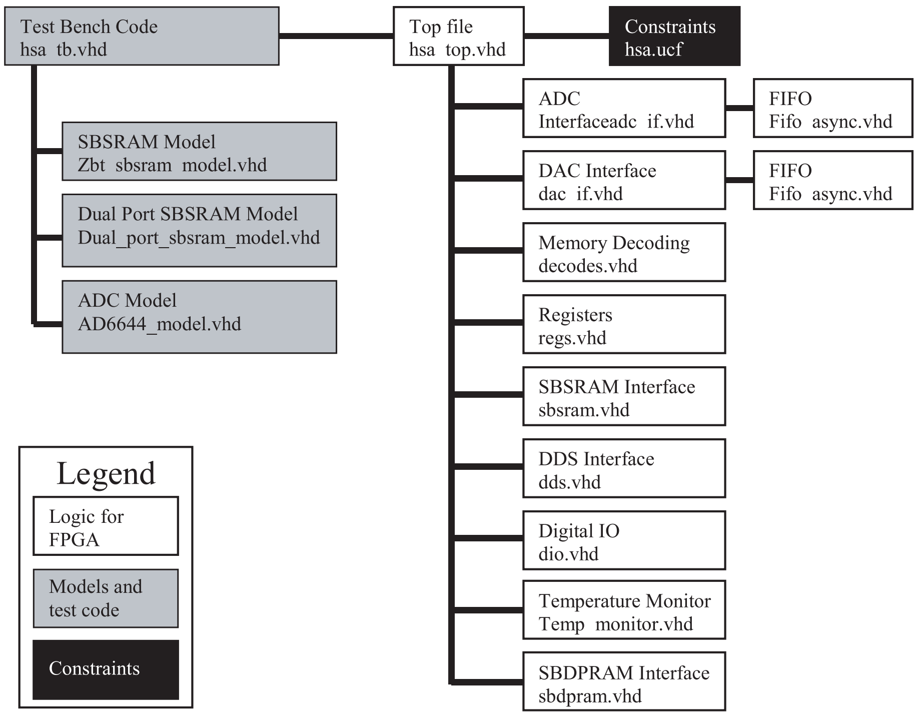
To burn the new .EXO file into the configuration flash memory on the HSA module, first copy the HSA.EXO file to the directory where the BURN_FL.OUT M6x executable resides. Run TERMINAL with the HSA module installed on the M6x, then run BURN_FL from TERMINAL. The program will automatically read the HSA.EXO file from the host disk, translate it to binary, and program the configuration flash ROM on the HSA module. The programming process depends on the size of the design being burned into the flash, but typically takes at least 2 minutes.

Once the programming process is finished, the Virtes device may be re-configured using the new information stored in the flash by running an M6x program which calls the **HSA_config_virtex_from_flash()** routine. The Virtex will NOT be reconfigured to the new data until this function is called.

Using the Framework Logic with Xilinx Foundation

The Framework logic may be recompiled within the Xilinx Foundation tools using the source codes and control file (UCF) provided. These files contain important controls for pin placement, pin type, and timing constraints. As part of the application design, the pins must remain fixed to match the HSA board. In some cases, such as the differential LVDS pairs and input pins, the source code contains further definitions for the pin output type that must not change so that the chip will not physically conflict with the other hardware.

The structure of the source code files is as shown in the following diagram.



The description of the Logic Files is shown in the following table.

File Name	Function
HSA_TOB.VHD	The top file for the synthesizable logic design.
DECODES.VHD	The memory map decoding.
ADC_IF.VHD	The ADC interface and FIFO.
REGS.VHD	The control registers and miscellany in the design.
DAC_IF.VHD	The DAC interface logic and FIFO.
DDS.VHD	The DDS interface and control logic.
TEMP_MONITOR.VHD	Temperature monitor interface.
SBSRAM.VHD	SBSRAM control and interface logic.
DIO.VHD	Digital IO control and interface logic.
FIFO_ASYNC.VHD	Logic for an asynchronous FIFO used in the DAC and ADC interfaces.
DP_SBSRAM.VHD	Dual port SBSRAM control and interface.
HSA_TB.VHD	The top level of the simulation test bench.
Ad6644_model.VHD	A simple model for the ADC converter.

File Name	Function
ZBT_SBSRAM_MODEL.VHD	A simple model for the SBSRAM memory.
DUAL_PORT_SBSRAM_MODEL.VHD	A simple model for the dual port SBSRAM memory.
Has.ucf	Constraints file for pins, prohibited pins, and timing.

Adding functionality to the Framework Logic

The framework logic is a starting point for the more advanced logic that will be your HSA application logic. It is suggested that you begin by simply recompiling this logic and verifying that you can recreate the framework logic as delivered. This will verify that you have all the libraries and FPGA compilation tools required to move ahead on your design.

Once you have successfully recompiled the logic, it is now possible to begin adding and replacing the simple logic with your application code. This is done by modifying the top VHDL to include your sub-functions, then modifying the test bench code to adequately stimulate your design.

Innovative Integration strongly recommends that you fully simulate your design before putting this logic into the HSA FPGA. This will not only save time in debugging, but could also prevent simple errors from causing serious damage to the module. A tool like ModelSim is generally required for this high density, complex logic design that give full visibility into the logic behavior prior to actual synthesis.

Many pre-written logic functions are available to assist in logic development from Xilinx and other vendors. These logic functions may be viewed at the Xilinx website (**`http://www.xilinx.com/ipcenter/index.htm`**). These logic functions include basic math, filters, FFTs, SBSRAM controllers, NCOs and a multitude of other functions that are useful in logic designing with the Xilinx Virtex FPGAs.

The logic source files are organized into three types: logic files for the FPGA, a constraint for the FPGA physical design, and the simulation test files.

The logic files are the framework files that are delivered in the FPGA logic from the factory. These logic files are synthesizable logic. The top file is HSA_TOP.VHD. This top file uses the components defined by the other logic files. Note that various Xilinx standard libraries are also used in the compilation process.

The constraints file must be used for the physical fitting into the FPGA. This file contains all the pin assignments, some basic timing constraints and some prohibited pins. This file should not be altered except to add timing constraints to the design.

The test files are used in the simulation and testing of the framework code. The testbench file is HSA_TB.VHD and it uses several components for testing that are defined by the other model files. These model files are very simple and are only for simple testing only. More complex models may be needed to adequately model more advanced uses. The testbench contains a set of simulation steps that exercise various functions on the framework logic for basic interface testing.

CHAPTER 11

MOT Module

Module Introduction

The Motion Controller Interface Module "MOT" provides the capability to interface with industry standard motion sensing and positioning equipment and act as a controller for up to four axes of movement per module. The MOT module provides four channels of quadrature position decoding with multiple independent index and home sensing inputs, four pulse outputs for controlling stepper motors, and four D/A outputs for driving DC servo motors. By combining the MOT module with the other OMNIBUS data acquisition and control modules, allows the DSP to bring together the data acquisition with motion control as a single integrated control system and eliminating the need for complex multi-card solutions.

Software design for the MOT module centers around a central timebase called the servo clock. All servo drive output electronics on the MOT module is driven by this single master clock source. This clock allows outputs to be synchronized and makes for a straightforward interrupt-driven real-time control program running on the host target processor card. Each new servo cycle automatically updates the drive electronics (stepper outputs or D/A channels, depending on the type of servo being designed) in hardware and generates an interrupt to the target processor on the target card. Typically this interrupt is used to service the control loop and its handler is where the control law is calculated for all axes and the next cycle's stepper, stepper direction, and D/A outputs are computed. These values are written during the routine to the various output control registers in time for the next control cycle to cause the hardware outputs to update to the new settings.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

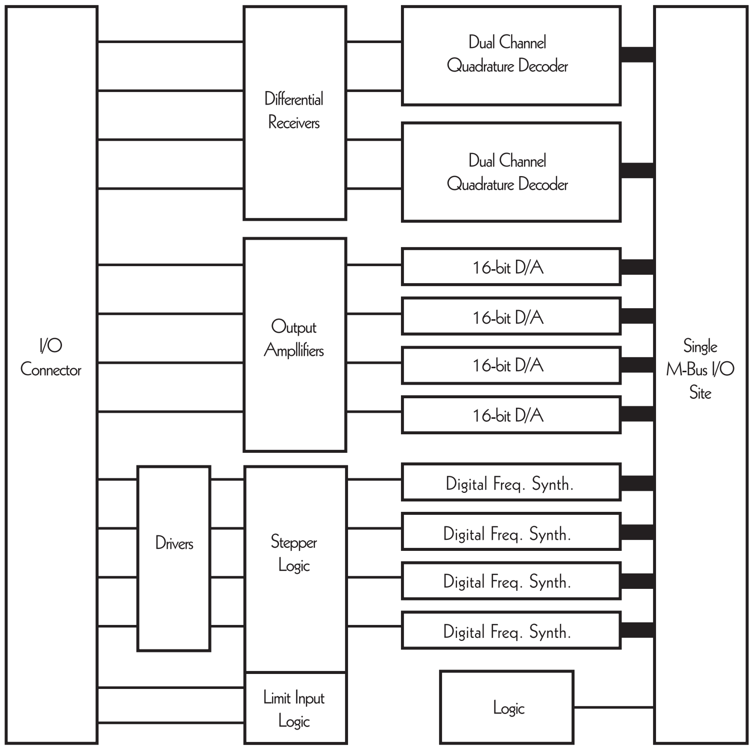


FIGURE 60. **MOT Block Diagram**

Bus Type:	Compatible with all I.I. products. Consumes one interrupt to host. Wait-state depends on host platform.	**Bipolar Zero Error:**	Trimmable
Power Requirements:	5 V @ 300 mA; +15 V @ 25 mA; -15 V @ 25 mA	**BiPolar Zero Error Drift:**	Trimmable
Physicals:	OMNIBUS mezzanine card; 2.0" X 4.6"	**Diff. Non-Linearity:**	+1 LSB
Servo Timebase:	AD9851 digital frequency synthesizer 0-16 MHz servo timebase on module in 0.01 Hz steps. All D/A and steppers are synchronized to the timebase DDS on MOT module.	**D/A Filter:**	Temperature Range 0 - 70 C; Filtering Output smoothing filter - single pole filter, 200 kHz rolloff (custom with cap/resistor change).
Digital Inputs:	Index Differential (RS422); Limit +/- are sensed using TTL inputs.	**Interface to DSP:**	Memory-mapped
D/A Converters:	Four Analog Devices AD669. Each D/A channel has independent filtering gain and trims.	**Stepper Motor Outputs:**	Four AD9850 digital frequency synthesizers; 0-16 MHz in 0.01 Hz steps. Open collector outputs; Memory-mapped to DSP.
Outputs:	Each output channel may be stepper or analog output, jumper selectable.	**Settling Time:**	13 us (no filtering) @ 20V step; 2.5 us for 1 LSB step settling to 0.0008%.
Resolution:	16-bit	**<u>Quadrature Decoders</u>**	
Output Range:	+/- 10 V custom w/resistor change.	**Number of Channels:**	4
Slew Rate:	15 V/us	**Max. Frequency:**	8 MHz
Update Rate:	200 kHz	**Input Type:**	Differential (RS422)
S/N Ratio:	0.0063% max.	**Counter Size**	24-bit
THD:	0.009% max.	**Position Capture Modes:**	Latch on external index or software.

The target Peripheral Library provides support for each of the MOT functions, and the following sections give descriptions of the support routines. Refer to the **motdac.h**, **quad.h** and **step.h** files located in the `c:/<target board>/Include/Target/Digital` director and **Omnibus.hlp or Zuma.hlp** Windows helpfile for complete details on each function.

Function Name	**Operation**
MOT_read_limit()	Read the current contents of the limit registers.
MOT_read_home()	Read the contents of the home register.
MOT_set_amp_enable()	Set the amplifier enable control bits.
MOT_load_QD_counter()	Load the quadrature counter.
MOT_latch_QD_counter()	Latch the quadrature counter.
MOT_read_QD_latch()	Read the quadrature latch.
MOT_clear_QD_counter()	Clear the quadrature counter.
MOT_reset_QD()	Reset the quadrature counter.
MOT_enable_normal_QD()	Enable normal counting mode.
MOT_enable_quadrature_QD()	Enable quadrature counting mode.
MOT_get_QD_flags()	Read the quadrature counter flags.
MOT_reset_QD_flags()	Reset the quadrature counter flags.
MOT_set_step_freq()	Set a stepper channel output frequency.
MOT_reset_step()	Reset a stepper channel.
MOT_setup_step()	Loads the four stepper setup channels.
MOT_set_servo_freq()	Program the servo timebase for the desired frequency.
MOT_correct_dac()	Adjusts the DAC signal to corrected results.
MOT_write_dac()	Latches the value of the specified D/A.
MOT_read_dac()	Reads last value written to DAC from shadow.
MOT_read_idrom()	Read identification information.
MOT_write_idrom()	Writes identification information from buffer.

TABLE 71. **C Language MOT Functions**

The MOT library functions can be divided into several groups:

1. Servo rate programming.
2. Quadrature counter management.
3. Stepper programming.
4. Limit/home switch readback.
5. Amplifier enable control.
6. D/A value load.
7. Identification readback.

Servo Rate Programming

The servo rate is programmed by a call to **MOT_set_servo_freq()**. This routine takes a frequency argument that it uses to calculate an appropriate period register value, which it then writes to the servo clock DDS device. The resulting output pulse train serves as an update clock for the various output functions on the MOT as well as an interrupt source for the target processor. The function returns an integer period register value, which is used by the application to program the individual stepper outputs (see below).

Quadrature Counter Management

The quadrature counters are configured and maintained by the quadrature counter management functions. These calls allow the target processor to initialize the operating mode of each counter channel, clear or load the counter, latch the current counter position, read the latched position, and read/reset the counter flag register.

In most motion control applications, counter initialization will occur once after power-up and at the applications start-up. The peripheral library supports running the counter in two of its standard modes: normal or quadrature (additional modes are possible: see the *target card Manual, "Hardware"* section for more details). Either the **MOT_enable_normal_QD()** or **MOT_enable_quadrature_QD()** calls must be made before using the counter.

Once the counter is initialized into a particular mode, it will count incoming pulses continuously, incrementing the counter for positive counts and decrementing the counter for negative counts. To read the counter's value, it must be latched (clocked into the output register of the quadrature counter). There are three ways a latch can be generated: by a software latch command (**MOT_latch_QD_counter()** call); in hardware by a simultaneous pulse on both the home and index inputs; or in hardware by an external index pulse (see the *target card Manual* for more information on hardware index and home detection). Once latched into the output buffer, the counter value may be read by a call to **MOT_read_QD_latch()**. The latch process is parallel and transparent, allowing the counter to continue counting incoming pulses while the processor reads the latched counter value.

The functions **MOT_clear_QD_counter()** and **MOT_load_QD_counter()** allow the processor to directly affect the counter's internal value. The **MOT_clear_QD_counter()** immediately clears the current count value to zero, while **MOT_load_QD_counter()** loads an arbitrary value into the counter register. As in the latch process, the transfers are transparent to the counting operation.

The quadrature counters implement a set of eight flag bits, which indicate certain overflow, underflow, count direction, and event conditions. The **MOT_get_QD_flags()** and **MOT_reset_QD_flags()** functions allow the program to read and reset the flags register.

Refer to the *target baseboard card Manual* for more details on registers.

Stepper Programming

The stepper programming routines allow the target processor to set the output rate and data format of the digital stepper outputs to suit the motion requirements and interface standards of the application. The stepper control logic consists of two parts: the DDS-based stepper frequency generator and the output formatting logic.

The base stepper frequencies are generated by a set of four AD9850 DDS synthesizer devices, which are capable of fine resolution adjustments that allow precise stepper pulse generation over a wide dynamic range. The DDS devices are nominally programmed by the application to produce a number of steps per servo cycle at the output. The library function **MOT_step_freq()** allows the step rate to be specified per channel as an integer number per cycle. The servo cycle frequency is defined to by passing in a frequency period word returned by the **MOT_servo_freq()** function (see above for details). **MOT_step_freq()** takes the pulses/cycle argument and multiplies it by the servo period value to generate a period value which it in turn writes to the DDS device for that channel.

Simply writing the period value to the DDS does not cause it to change its frequency output: the servo pulse actually causes the update on a servo cycle boundary. Application software needs to get the next cycle's stepper frequency written to the DDS via **MOT_step_freq()** before the next servo cycle in order to ensure that the next cycle's frequency is output correctly.

The DDS's digital stepper output is fed by the MOT hardware into the stepper formatting logic, which is used to alter the pulse sense and generate various formats suitable for stepper amplifiers from different manufacturers. Two basic formats are supported, with two inversion options and in two different directions for a total of eight different possible stepper motor outputs (see the *target card Manual* for a description of the possible formats). The **MOT_setup_step()** function is used to control the format as well as the direction of the formatting logic, and is called once at the initialization of the card to set the format bits in the stepper control register. These formatting bits are typically held constant through the run of the application, as it is unlikely that format changes would be required under any single program run. The function, however, can be called repeatedly if format changes are required during the run.

The **MOT_setup_step()** function also enables and disables the stepper fail-safe features of the MOT module. When fail-safe mode is on for a particular axis and a limit switch is active, stepper pulse outputs to the amplifiers are disabled in the direction corresponding to the limit switch. This prohibits further movement through the end of travel area denoted by the limit switch. Stepper outputs in the opposite direction are not prohibited and application software may use opposite direction stepper pulses to "back" the system to a legal movement area.

Limit/Home Switch Readback

The MOT libraries provide support for reading the current values of the limit and home switch inputs on each channel. The **MOT_read_limit()** function returns the values of the limit switch inputs as an eight-bit number, with axis zero's limit switches in bits zero (plus limit) and one (minus limit), axis one's switches in bits two (plus limit) and three (minus limit), etc.

MOT_read_home() reads the current status of the home register bits, which will go true (high) when a home event has occurred on the accompanying home input pins. The four least significant bits contain the current register value, which is cleared in hardware by the **MOT_read_home()** call. *Home Finding* - Precise home finding is achieved with a hardware counter latched by a simultaneous pulse on both the home and index inputs. The value can be read with **MOT_read_QD_latch()**.

Amplifier Enable Control

The TTL amplifier enable outputs for each axis are supported by the **MOT_set_amp_enable()** function. This call sets the current amplifier enable outputs to the argument value, allowing software to enable/disable TTL controllable motor amplifiers.

D/A Value Load

The D/A converter's input latches may be loaded using the **MOT_write_dac()** function. The function writes a 16-bit unsigned value to the specified D/A device, which will become the voltage output from the D/A at the beginning of the next servo cycle.

Identification Readback

The **MOT_read_idrom()** function is used to read the identification ROM on the MOT to check its identity and revision level. The function fills out an **MOT_id** structure with the information stored in the ID ROM on the module. The **MOT_id** structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string "MOT").
2. Module revision level.
3. Number of axes populated on the module.
4. Checksum.

This data is preprogrammed at the factory and should not be altered by the user.

Example DSP Target Programs for the MOT Module

The **t_mot.c** program located in the EXAMPLES\TARGET\MOT subdirectory is used to test the MOT module hardware via the following functions:

Function	Description
void test_dac(void);	Verifies performance of the onboard D/A converters.
void test_home_index(void);	Verifies performance of the external TTL homing inputs
void test_quad(void);	Verifies the four channels of quadrature encoder input.
void test_limit(void);	Verifies external TTL limit input channels.
void test_amp_enable(void);	Verifies external TTL amplifier enable outputs
void test_steppers(void);	Verifies performance of pulse and direction stepper outputs. Four channels.
void test_id(unsigned int);	Verifies proper operation of the onboard ID ROM.

Memory Mapping

The MOT module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** file included with the baseboard Development Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
9850 Channel 0 Write	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1
9850 Channel 1 Write	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3
9850 Channel 2 Write	IOMOD0 + 0x5	IOMOD4 + 0x5	IOMOD8 + 0x5
9850 Channel 3 Write	IOMOD0 + 0x7	IOMOD4 + 0x7	IOMOD8 + 0x7
Home/Index Status and Reset	IOMOD0 + 0x8	IOMOD4 + 0x8	IOMOD8 + 0x8
Limit Status 0 (Ch 0-1)	IOMOD0 + 0x9	IOMOD4 + 0x9	IOMOD8 + 0x9
Limit Status 1 (Ch 2-3)	IOMOD0 + 0xA	IOMOD4 + 0xA	IOMOD8 + 0xA
Stepper Channel 0 Setup	IOMOD0 + 0xB	IOMOD4 + 0xB	IOMOD8 + 0xB
Stepper Channel 1 Setup	IOMOD0 + 0xC	IOMOD4 + 0xC	IOMOD8 + 0xC
Stepper Channel 2 Setup	IOMOD0 + 0xD	IOMOD4 + 0xD	IOMOD8 + 0xD
Stepper Channel 3 Setup	IOMOD0 + 0xE	IOMOD4 + 0xE	IOMOD8 + 0xE
Amplifier Enable	IOMOD0 + 0xF	IOMOD4 + 0xF	IOMOD8 + 0xF
D/A Channel 0 Write	IOMOD0 + 0x10	IOMOD4 + 0x10	IOMOD8 + 0x10
D/A Channel 1 Write	IOMOD0 + 0x11	IOMOD4 + 0x11	IOMOD8 + 0x11
D/A Channel 2 Write	IOMOD0 + 0x12	IOMOD4 + 0x12	IOMOD8 + 0x12

Function	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
D/A Channel 3 Write	IOMOD0 + 0x13	IOMOD4 + 0x13	IOMOD8 + 0x13
9850 Servo Timebase Reset	IOMOD0 + 0x14	IOMOD4 + 0x14	IOMOD8 + 0x14
9850 Servo Timebase Write	IOMOD0 + 0x15	IOMOD4 + 0x15	IOMOD8 + 0x15
9850 Servo Timebase Update	IOMOD0 + 0x16	IOMOD4 + 0x16	IOMOD8 + 0x16
7266 Channel 0/1	IOMOD0 + 0x20	IOMOD4 + 0x20	IOMOD8 + 0x20
7266 Channel 2/3	IOMOD0 + 0x30	IOMOD4 + 0x30	IOMOD8 + 0x30
IDROM	IOMOD3	IOMOD7	IOMOD11

TABLE 72. **MOT Memory Map**

Interrupt Usage

The MOT module requires one host processor interrupt input for servo loop interrupts. The module generates one interrupt to the host at the beginning of each servo cycle (i.e. at the rate programmed into the servo cycle timebase). This interrupt is typically used to interface with the hardware and service the control law by gathering position information from the quadrature decoders and programming new output rates into the stepper logic or D/A converters.

The MOT module is capable of generating interrupts to the host processor on external interrupt input zero (for a module installed in I/O bus slot 0) or two (for a module installed in I/O bus slot 1). The interrupt is generated at each servo cycle (i.e. a 10 kHz servo cycle rate will cause a 10 kHz interrupt rate to the processor).

Pin Connector I/O

The MOT output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* section for additional details on how to connect to the MOT's I/O pins.

MOT Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50 Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1	100	75	25	1		Channel 0 Step+/DAC Output
2	50	25	24	2		Channel 0 Step-/Direction Output
3	99	74	23	3		Channel 1 Step+/DAC Output

MOT Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50 Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
4	49	24	22	4		Channel 1 Step-/Direction Output
5	98	73	21	5		Channel 2 Step+/DAC Output
6	48	23	20	6		Channel 2 Step-/Direction Output
7	97	72	19	7		Channel 3 Step+/DAC Output
8	47	22	18	8		Channel 3 Step-/Direction Output
9	96	71	17	9		Channel 0 Limit 0 Input
10	46	21	16	10		Channel 0 Limit 1 Input
11	95	70	15	11		Channel 1 Limit 0 Input
12	45	20	14	12		Channel 1 Limit 1 Input
13	94	69	13	13		Channel 2 Limit 0 Input
14	44	19	12	14		Channel 2 Limit 1 Input
15	93	68	11	15		Channel 3 Limit 0 Input
16	43	18	10	16		Channel 3 Limit 1 Input
17	92	67	9	17		Channel 0 Home Input
18	42	17	8	18		Channel 1 Home Input
19	91	66	7	19		Channel 2 Home Input
20	41	16	6	20		Channel 3 Home Input
21	90	65	5	21		Channel 0 Quadrature A+ Input
22	40	15	4	22		Channel 0 Quadrature A- Input
23	89	64	3	23		Channel 0 Quadrature B+ Input
24	39	14	2	24		Channel 0 Quadrature B- Input
25	88	63	1	25		Channel 0 Index+ Input
26	38	13	50	26		Channel 0 Index- Input
27	87	62	49	27		Channel 1 Quadrature A+ Input
28	37	12	48	28		Channel 1 Quadrature A- Input
29	86	61	47	29		Channel 1 Quadrature B+ Input
30	36	11	46	30		Channel 1 Quadrature B- Input
31	85	60	45	31		Channel 1 Index+ Input
32	35	10	44	32		Channel 1 Index- Input
33	84	59	43	33		Channel 2 Quadrature A+ Input
34	34	9	42	34		Channel 2 Quadrature A- Input
35	83	58	41	35		Channel 2 Quadrature B+ Input
36	33	8	40	36	1	Channel 2 Quadrature B- Input
37	82	57	39	37	9	Channel 2 Index+ Input
38	32	7	38	38	2	Channel 2 Index- Input
39	81	56	37	39	10	Channel 3 Quadrature A+ Input
40	31	6	36	40	3	Channel 3 Quadrature A- Input
41	80	55	35	41	11	Channel 3 Quadrature B+ Input
42	30	5	34	42	4	Channel 3 Quadrature B- Input
43	79	54	33	43	12	Channel 3 Index+ Input
44	29	4	32	44	5	Channel 3 Index- Input
45	78	53	31	45	13	Amplifier 0 Enable Output
46	28	3	30	46	6	Amplifier 2 Enable Output
47	77	52	29	47	14	Amplifier 1 Enable Output
48	27	2	28	48	7	Amplifier 3 Enable Output
49	76	51	27	49	15	Analog Ground
50	26	1	26	50	8	Reserved

TABLE 73. **MOT I/O Connector Pinout**

Functions

Servo Timebase

The MOT module uses a single AD9850 direct digital synthesizer (DDS) device to implement a master servo clock timebase. This timebase serves as the update clock for all four stepper motors and D/A outputs, in addition to being able to generate an interrupt back to the host processor. This allows motion control software to operate as an efficient interrupt driven routine. It can retrieves the current position information from the quadrature decoders, calculates new motion rates for each active channel, and programs the stepper and D/A outputs for the next cycle. Motion outputs latched to the output hardware are updated automatically on each servo clock cycle, which means that the new output values programmed into the D/As or steppers will become active at the beginning of the next cycle.

The servo timebase is capable of 0.014 Hz resolution with an output frequency range of DC to 30 MHz. This wide dynamic range allows for almost infinite control of the servo cycle frequency.

The AD9850 device is accessed through a set of three registers, which allow complete control over reset, control register loads, and timebase output updates. Each channel has a reset control register that allows the I/O bus host processor to control hardware reset on the DDS. The following table details the control sense of the AD9850 reset control register.

Register Value	Function
0	AD9850 normal mode
1	AD9850 in reset

TABLE 74. **MOT AD9850 DDS Reset Control Register Values**

The servo's write control register is used to transfer new output frequency information to the AD9850. This is done as a sequence of byte writes of coefficient data on the least significant byte of the host processor's data bus. Once the sequence has been completed, the DDS's output frequency may be updated by an access (read or write) to the AD9850 update register. Please see the AD9850 data sheet for more information about the content of the frequency coefficient data.

Quadrature Decoder/Counters

The MOT includes two LS7266 dual quadrature decoder/counter devices, which provide a total of four quadrature decoder channels. These devices are capable of receiving both quadrature and count plus direction pulse inputs and provide one 24-bit signed counter accumulation register per channel as well as various other preset and control registers.

The LS7266 devices are accessed through their memory map addresses as a set of control registers defined by the LS7266 data sheet. The device's register decoding also includes data bits 7, 6, and 5 on the byte-wide interface (see the LS7266 data sheet for details and a register map).

Each of the LS7266 devices' XLCNTR/LOL inputs is tied to an independent channel home/index latch signal generated by onboard logic whenever a home or index event occurs. This allows each counter channel to independently latch or load the counter value upon a home or index event, which in turn allows a host processor's home/index interrupt event handler to read back a hardware latched or loaded position. This is a typical method used in motion control to perform precise homing operations or to set accurate index offsets.

NOTE: The LS7266 XRCNTR/XABG, XBW, and XCY inputs are unused in the MOT module design.

Quadrature Decoder Inputs

The MOT module expects to receive a standard A/B quadrature pair when used in quadrature counting mode, or a count/direction signal pair (with count pulses present on the A input and direction present on the B input).

Both differential and single-ended quadrature input signals may be used with the MOT module. The pinout notes plus and minus inputs for each of the quadrature signal channels: the positive differential signal is tied to the plus input, and the negative differential signal is tied to the minus input. Single-ended signals are always tied to the plus input, while the minus input is left unconnected.

The minus inputs are biased to 2.5V using 1K resistors to +5V and to ground on each input. This allows the user to leave the minus inputs disconnected when driving only a single ended quadrature signal into the plus inputs (the bias resistor holds the minus inputs at a midpoint voltage to ensure that the differential receivers see the plus input signal swing).

Home, Index, Limit, and External Index Inputs

The MOT module includes 16 channels of home, limit, and index inputs for monitoring absolute position information events from position sensors on a motion platform. Each input provides digital event notification, which allows the I/O bus host processor to monitor when motion axes have reached a relative home (zero) position or a maximum safe limit stop, or when rotational or linear encoders travel past index positions.

The home inputs are single-ended TTL and falling edge sensitive (i.e. the signals are normally high and fall low at the beginning of a home event). Falling edges on the home input pins are latched into the internal logic. The latched value may be read back by accessing the home/limit status/reset register. The least significant four bits of the register hold the home/index latch values for the four home inputs (see below for more information on the index inputs). A high bit indicates a home/index event occurred on the corresponding channel since the last read of the register, while a low value indicates that no event occurred. A read of the register will clear all bits.

In addition to being latched in the home/index status/reset control register, the home inputs also serve to qualify the load/latch pulse to the quadrature decoders. A simultaneous low on both the home and index inputs for a particular channel causes an active load/latch pulse to be generated to the corresponding quadrature decoder. This facilitates automatic latching of the current position when home and index events occur simultaneously (which is often useful in determining precise home positioning).

The index inputs may be operated in either differential or single-ended mode and use TTL signal levels. The pinout includes plus and minus inputs for each channel: when connecting in differential mode both inputs are driven, while in single ended mode the plus input is driven and the minus input left unconnected. The index input is latched and status monitored as described above for the home inputs.

The limit inputs are single-ended TTL, with two limit inputs provided per axes. The limit inputs are implemented as a set of digital input pins, accessible to the I/O bus host processor by reading the limit status registers. The least significant four bits of each register return the corresponding limit input pin values (the following table gives a breakdown of register readback contents).

Limit Register	Register Bit Position	Function
0	0	Channel 0 Limit 0
	1	Channel 0 Limit 1
	2	Channel 1 Limit 0
	3	Channel 1 Limit 1
1	0	Channel 2 Limit 0
	1	Channel 2 Limit 1
	2	Channel 3 Limit 0
	3	Channel 3 Limit 1

TABLE 75. MOT Limit Status Register Format

The external index inputs provide an additional means to trigger a quadrature decoder latch event. The external index inputs are provided on a separate 2mm single in-line header on the MOT module, and are single-ended TTL active low. Each of the index input is capable of asserting the corresponding quadrature channel's LCNTR/LOL input, independent of the regular index and home inputs. The header is labeled (JP31), and the following table gives the pinout (the square solder pad on the module marks pin 1).

Pin Number	Function
1	DGND
2	External Index 3
3	External Index 2
4	External Index 1
5	External Index 0

TABLE 76. MOT External Index Input Pinout

The home, limit, and external index inputs are each pulled up to +5V through a 1K ohm resistor. Each index channel's differential minus input is pulled to +5V and to ground through 1K ohm resistors, while the plus input is pulled up to +5V through a 1K ohm resistor.

Stepper Motor Timebases

The MOT module implements four stepper motor digital outputs which are based on AD9850 DDS devices. Each channel has an independent DDS, which acts as a square wave generator capable of outputting continuous pulse trains at a high frequency resolution. The devices are capable of 0.014 Hz resolution with an output frequency range of DC to 30 MHz. This software programmable frequency makes very precise position control possible, even on the fastest of motion servo systems.

The DDS write control register is used to transfer new output frequency information to the AD9850. This is done as a sequence of byte writes of coefficient data on the least significant byte of the host processor's data bus. Please see the AD9850 data sheet for more information about the content of the frequency coefficient data.

DDS output frequency updates are controlled by the servo timebase. At the beginning of each servo cycle, a reset pulse followed by an update pulse is generated to each DDS, which causes its output frequency to change to the current value programmed into the device's frequency control register (done previously through the write control register discussed above). The output frequency does not update until this servo timebase pulse is received: no software interface is provided to allow application programs to perform a manual update. The servo timebase must be programmed to the expected servo rate in order for output updates to be performed.

Stepper Motor Outputs

The stepper motor output circuitry on each channel takes the AD9850 square wave as input and optionally modifies the polarity and format for compatibility with stepper motor amplifiers from different manufacturers. Direction control and a fail-safe option are also programmable for each channel.

Two output signals are provided for each stepper motor control axis, and these can be programmed to output stepper control data in several different formats. Two basic formats are provided, with two polarity options for each type:

1. Step+/Step- - the stepper motor output is presented with positive direction pulses appearing on one output and negative direction pulses appearing on the other.

2. Step/Direction - the stepper motor output is presented with continuous step counts on one output and a direction control voltage (high for one direction, low for the other) on the other output.

The format, direction, polarity, and fail-safe mode for each channel are independently programmable with the stepper setup registers. The following table gives the setup register definitions.

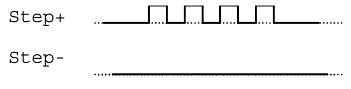
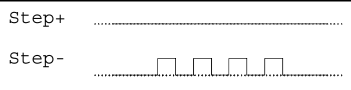
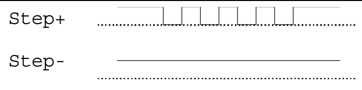
Register Bit Position	Function	Value	Effect
0	Format Select	0	Step+/Step- output format
		1	Step/Direction output format
1	Stepper Direction	0	Stepper output is in negative direction
		1	Stepper output is in positive direction
2	Output Polarity	0	Stepper output is active low
		1	Stepper output is active high
3	Fail-Safe Enable/Disable	0	Fail-safe disabled
		1	Fail-safe enabled

TABLE 77. **MOT Stepper Output Setup Register**

Please Note: The stepper outputs are not controlled on a system power up and may settle to either TTL voltage level. Please take this into account when designing external hardware so as to provide appropriate power up control to offboard circuitry. The amplifier enable controls (see below) are always reset (set low) on power up and may be used to control external motor amplifiers which accept TTL inputs.

Stepper Motor Output Formats and Polarity Options

The following table give graphical representations of the possible output formats, along with the required setup register value to achieve each output type.

Format	Polarity	Direction	Setup Register Value	Stepper Pin Outputs
Step+/Step-	Active high	Positive	X010	
	Active high	Negative	X000	
	Active low	Positive	X110	

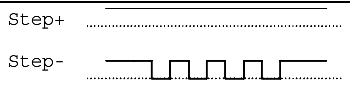
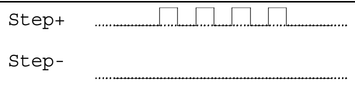
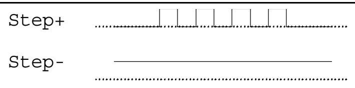
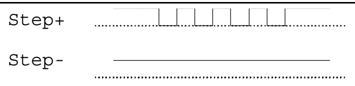
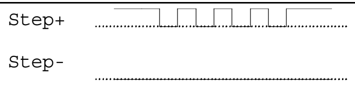
Format	Polarity	Direction	Setup Register Value	Stepper Pin Outputs
	Active low	Negative	X100	
Step/Direction	Active high	Positive	X011	
	Active high	Negative	X001	
	Active low	Positive	X111	
	Active low	Negative	X101	

TABLE 78. MOT Stepper Setup Options

Stepper Output Fail-Safe

The stepper outputs also have optional independently programmable fail-safe functions, which allow the stepper outputs to be shut off automatically under certain circumstances. The fail-safe function is keyed to the limit switch inputs and will disable stepper pulses unidirectionally if a limit switch is tripped and the fail-safe function is enabled, thus preventing continued mechanical movement in the direction of the limit switch. The fail-safe does not limit further movement in the opposite direction, so the operator or servo controller is free to use the stepper output in the opposite direction to return the motion stage to a safe area of motion. Although the limit switches should otherwise be monitored by control software running on the host CPU in order to avoid out of limit operation, the fail-safe feature provides a hardware based, near instant cutoff response.

Note: the fail-safe system is keyed to the polarity of the stepper outputs, so it is important to make sure that limit switches have been properly installed and matched with the positive versus negative mechanical polarity of the motion stage. The fail-safe disables positive stepper pulses when the positive limit switch (limit 0) is engaged, and disables negative stepper pulses when the negative limit switch (limit 1) is engaged. The fail-safe feature is axis independent, so movement on other axes is possible in both directions even if one or more axes are fail-safe limited.

Also note: the fail-safe system ONLY applies to stepper motor outputs, and does NOT apply to the analog output circuitry. DC servo control algorithms cannot rely on the stepper motor fail-safe system as a safety backup.

Analog Outputs

The MOT module also provides four 16-bit D/A converters for generating output drive signals for use in DC servomotor applications. The AD669 devices have fast conversion rates (100-200 kHz max) and high resolution for demanding positioning systems.

The D/A interface to the I/O bus host processor provides one data latch register per channel for each D/A converter. Digital data to be converted into output voltage is written to each of these locations, with voltage output updates triggered by the servo timebase. The output of each D/A channel is +-10V bipolar, with an inverting phase (i.e. a zero written to the D/A causes a +10V output, and a 65535 written to the D/A causes a -10V output). Output updates may only be triggered by the servo timebase and software triggered updates are not supported.

Each D/A channel includes an output amplifier with a gain factory set to one. The amplifier's gain may be altered to allow output ranges other than the default +-10V range, if required. (The AD669 devices' output is nominally +-10V). The following diagram shows the topology of the amplifier and the table gives the resistor gain pair reference designators for each channel. The output amplifiers run off +-15V rails, providing a maximum usable output swing over temperature of +-13.5V.

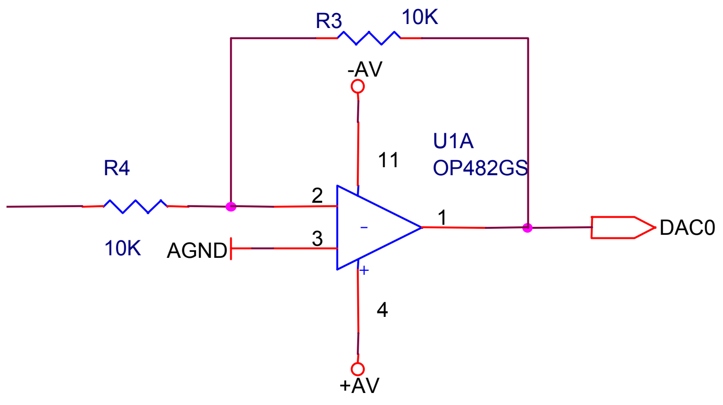


FIGURE 61. **MOT D/A Output Amplifier**

Output Channel	Gain Equation
0	-R3/R4
1	-R7/R8
2	-R11/R12
3	-R15/R16

TABLE 79. **MOT D/A Output Amplifier Gain Equations**

Please Note: The D/A outputs are not controlled on system power up and may settle to any level within the nominal output range of +-10V. Please take this into account when designing external hardware so as to provide appropriate power up control to offboard circuitry. The amplifier enable controls (see below) are always reset (set low) on power up and may be used to control external motor amplifiers which accept TTL inputs.

D/A Output Trim

Each D/A output provides trim potentiometers for gain and offset which can provide approximately 200 mV of adjustment range for system level trimming. The following calibration method may be used in the field to null offset and gain errors.

1. Set the D/A output for +10.0V output by programming all zeros into the data latch. Adjust the offset trimmer (see below) until +10V is seen at the output.

2. Set the D/A output for -10.0V output by programming all ones (i.e. 0xffff) into the data latch. Adjust the gain trimmer (see below) until -10V is seen at the output.

3. Set the D/A for 0V output by programming 0x8000 into the data latch. Readjust the offset trimmer until 0V is seen at the output.

Caution: The DAC data latch values (shown above) are not the same as the last argument in **MOT_write_dac (site, chn, value)** because this function calls **MOT_correct_dac()** in **MOT.H**. See corresponding values in table below.

DAC Output	Data Latch Value	Last Argument in MOT_write_dac()
+10	0x7FFF	0x0000
-10V	0x8000	0xFFFF
0V	0x0000 or 0xFFFF	0x8000

The following table gives the locations of the trim potentiometers for each D/A channel.

D/A Channel	Offset Trim Potentiometer	Gain Trim Potentiometer
0	R5	R6
1	R9	R10
2	R13	R14
3	R17	R18

TABLE 80. **MOT D/A Trim Potentiometers**

Output Mode Selection

The MOT module provides four jumpers, which are used to select the motor control output signal type for each axis. Each jumper allows the user to select between analog outputs (generated by the D/A circuitry) or stepper motor outputs (from the stepper motor output logic). The following table gives jumper settings for each channel.

Channel	Jumper Reference Designator	Position	Function
0	JP25	1-2	Stepper Output
		2-3	Analog Output
1	JP26	1-2	Stepper Output
		2-3	Analog Output
2	JP27	1-2	Stepper Output
		2-3	Analog Output
3	JP28	1-2	Stepper Output
		2-3	Analog Output

TABLE 81. **MOT Motor Control Output Jumper Selection**

Amplifier Enable Control

Output amplifier control is supported through the use of the amplifier enable control register. This register allows the MOT to generate four single-bit TTL outputs (one per channel) which can be used for external motor amplifier/driver control.

The amplifier enable register is programmed by writing to the register with the least significant four bits set to the desired output bit pattern. The values are latched from the I/O bus and held until the next write to the amplifier enable control register. The bit values from the bus are driven directly out to the amplifier enable pins on the I/O connector. The register is cleared (outputs driven low) on power up or after a host hardware reset.

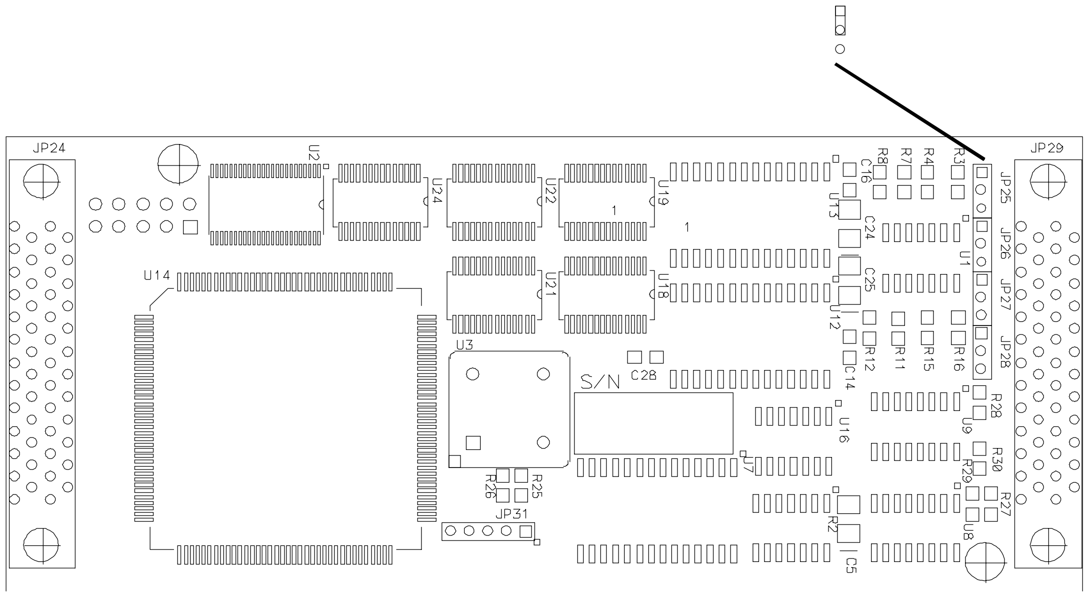
Electrical Isolation

Caution, Please Note: The MOT module does NOT provide electrical isolation on any of its I/O pins. Industrial motion control installations often involve high voltages and inductively switched loads, which can cause voltage and/or current transients on the control busses. Exceeding the positive or negative voltage rails or sinking/sourcing high currents can cause damage to the MOT module and host processor card. It is the user's responsibility to provide adequate isolation from external transients and high input/output currents. Never connect the MOT I/O pins to external hardware with the host system powered down. All external hardware to which the MOT is connected must be powered on and off simultaneous with the host system, or CMOS latchup damage to the OMNIBUS host and MOT module can result.

Factory Jumper Settings

JP25, 26, 27, 28 - Motor Control Output Type (Ch 0, 1, 2, 3)

1-2 Stepper Output
2-3 D/A Output



CHAPTER 12

RF Module

Module Introduction

The RF module provides the target card with two channels of high speed 65 MHz, 12-bit resolution analog input conversion (A/D) and (D/A). The input signal processing chain consists of 50 ohm input termination, a high speed low distortion input amp, a programmable gain amp giving a signal amplification range of -14 dB to +34 dB, followed by a 7th order low-pass filter to the A/D. The two's complement data from the A/D converters clock into 1024-sample FIFOs, which can be configured to interrupt the host card at programmable levels.

The RF module also provides the target card with two channels of 12-bit resolution digital output conversion (D/A). Each of the output channels includes a 7th order low-pass filter, an output buffer amp, and has a single-ended output matched to 50 ohms. The D/A's are directly fed by a 1024-sample FIFO and triggered by the onboard DDS or an external clock source. The D/A FIFO levels are monitored in a FIFO status register and can trigger interrupts to the host card.

The RF module has it own on board DDS (Direct Digital Synthesizer) timebase, driven by a high stability temperature compensated oscillator. The DDS may be used as a programmable sample rate generator from 0 to 80 MHz with a 0.02 Hz resolution. Digital gain and offset corrections are done in the FPGA during realtime for both the A/D and D/A channels. The RF module also has the ability to stack the data samples in order to maximize the use of its 32-bit word length and increase the overall sample rate. This means two samples of a 12-bit channel can be stacked on the 32-bit word to double the bandwidth or four samples of 8 significant bits can be stacked to quadruple the read/write rates.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

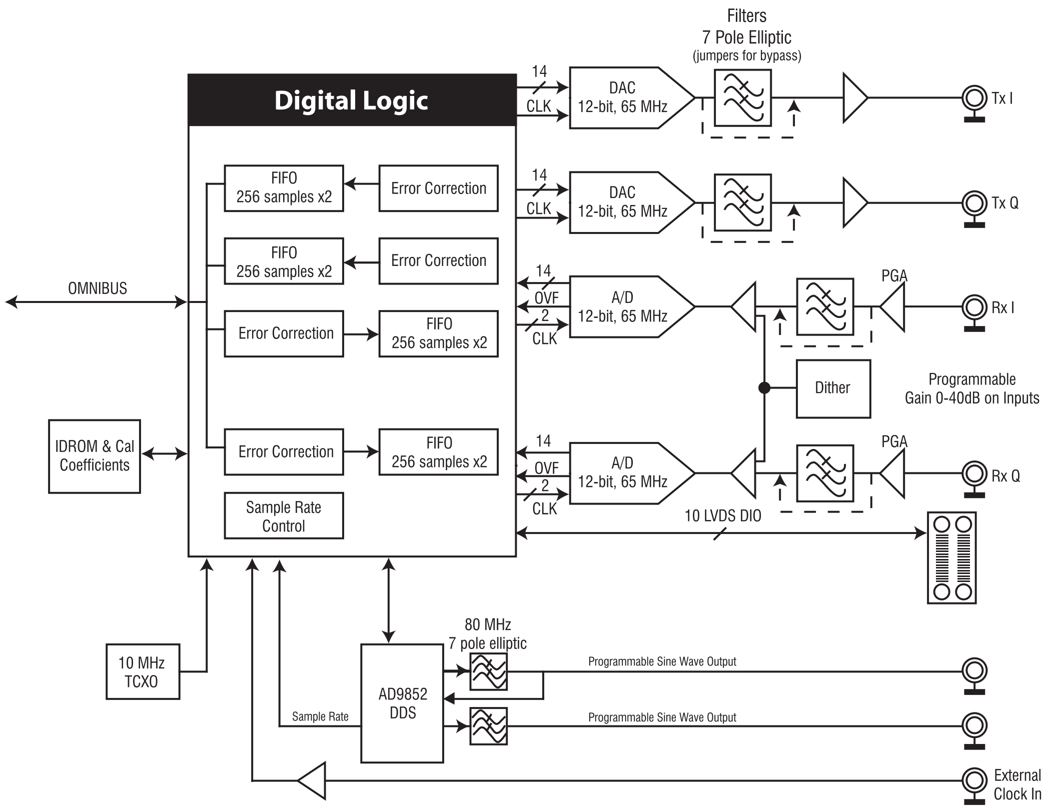


FIGURE 62. RF Block Diagram

Bus Type:	Compatible with all OMNIBUS host products; Consumes one interrupt to host baseboard.	**Conversion Trigger Sources:**	On board DDS or External clock; Optional dither on A/D for improved S/N.
Power Requirements:	2.0 W	**D/A Converter:**	Analog Devices AD9765 converters.
Physicals:	OMNIBUS mezzanine card; 2.0” X 4.6”	**Resolution:**	12-bit
A/D Converters	2 Analog Devices AD9226 converters.	**Output Range:**	+/- 1V
Resolution:	12-bit	**Settling Time:**	35 ns
Update Rate:	65 MHz. (Maximum)	**Dynamic Range:**	72 dB
Analog Input Type:	Single ended to 50 ohm SMB connector.	**THD:**	0.01%
Analog Input Range:	+/- 1V at 0 db; +40 db with programmable gain amplifier.	**Offset Error:**	Trimmable on each channel - factory calibrated.
Analog Input Impedance:	50 ohm	**Gain Error:**	Trimmable on each channel - factory calibrated.
Input Filter Characteristics:	Lowpass or Bandpass 7 pole -3 dB point @ 12 MHz; Jumperable	**Interface to DSP:**	Memory mapped 32-bit result, programmable via Xilinx FPGA
SNR:	>64 dB (without variable gain) >73 dB (with variable gain)	**DDS - AD9852:**	200 MHz input with 40-bit resolution in frequency max. DDS maximum output rate is 80 MHz.
SINAD:	45 dB (with variable gain)	**TCXO:**	5 ppm TCXO (10 MHz) onboard
THD:	<0.01%	**Digital I/O:**	10 pairs LVDS I/O
Interface to Host Card:	Memory mapped configurable 32-bit result.		

Memory Mapping

The RF module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** or the **rf.h** file included with the host board's Development Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	**Read/Write**	**OMNIBUS Slot 0 Address**	**OMNIBUS Slot 1 Address**	**OMNIBUS Slot 2 Address**
Module data, read A/D data, write D/A data	R/W	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0
Run Control Register	W	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1
A/D Control Register	R/W	IOMOD0 + 0x2	IOMOD4 + 0x2	IOMOD8 + 0x2
D/A Control Register	R/W	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3
A/D Clock Configuration Register	R/W	IOMOD0 + 0x4	IOMOD4 + 0x4	IOMOD8 + 0x4
D/A Clock Configuration Register	R/W	IOMOD0 + 0x5	IOMOD4 + 0x5	IOMOD8 + 0x5
DDS Control Register	R/W	IOMOD0 + 0x8	IOMOD4 + 0x8	IOMOD8 + 0x8
DDS Data Register	R/W	IOMOD0 + 0x800	IOMOD4 + 0x800	IOMOD8 + 0x800
A/D Mode Register	R/W	IOMOD0 + 0xA	IOMOD4 + 0xA	IOMOD8 + 0xA
D/A Mode Register	R/W	IOMOD0 + 0xB	IOMOD4 + 0xB	IOMOD8 + 0xB
A/D Gain Amp Control Register	W	IOMOD0 + 0xC	IOMOD4 + 0xC	IOMOD8 + 0xC
A/D 0 Gain Register	R/W	IOMOD1 + 0x0	IOMOD5 + 0x0	IOMOD9 + 0x0
A/D 1 Gain Register	R/W	IOMOD1 + 0x1	IOMOD5 + 0x1	IOMOD9 + 0x1
D/A 0 Gain Register	R/W	IOMOD1 + 0x2	IOMOD5 + 0x2	IOMOD9 + 0x2
D/A 1 Gain Register	R/W	IOMOD1 + 0x3	IOMOD5 + 0x3	IOMOD9 + 0x3
A/D 0 Offset Register	R/W	IOMOD2 + 0x0	IOMOD6 + 0x0	IOMOD10 + 0x0
A/D 1 Offset Register	R/W	IOMOD2 + 0x1	IOMOD6 + 0x1	IOMOD10 + 0x1
D/A 0 Offset Register	R/W	IOMOD2 + 0x2	IOMOD6 + 0x2	IOMOD10 + 0x2
D/A 1 Offset Register	R/W	IOMOD2 + 0x3	IOMOD6 + 0x3	IOMOD10 + 0x3
ID ROM	R/W	IOMOD3 + 0x0	IOMOD7 + 0x0	IOMOD11 + 0x0
ID ROM sclk	W	IOMOD3 + 0x1	IOMOD7 + 0x1	IOMOD11 + 0x1
SROM Programming Register	R/W	IOMOD3 + 0x4	IOMOD7 + 0x4	IOMOD11 + 0x4
SROM Programming Enable Register	W	IOMOD3 + 0x5	IOMOD7 + 0x5	IOMOD11 + 0x5

TABLE 82. RF Memory Map

Interrupt Usage

The RF module has a single interrupt output that indicates when data is available to be read from the FIFO. The interrupt can be programmed to trigger on the FIFO not empty condition or when the FIFO exceeds a set threshold condition. This feature allows the programmer to pace the data retrieval from the module based upon the expected data rate.

The interrupt enable and mode selection is controlled by the control register, while the threshold value (if used) is controlled by the FIFO threshold register. See below for details on programming these features.

Pin Connector I/O

The RF output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* for additional details on how to connect to the RF's I/O pins.

RF Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50 Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1..23	40..50, 89..100	15..25, 64..75	3..25	1..23		Reserved
24	39	14	2	24		External Gate
25	88	63	1	25		Ground
26	38	13	50	26		Ground
27..42	23..37, 80..87	5..12, 55..62	34..49	27..42	3, 4, 10, 11	Reserved
43	79	54	33	43	12	D/A Sync Out (positive)
44	29	4	32	44	5	D/A Sync Out (negative)
45	78	53	31	45	13	
46	28	3	30	46	6	
47	77	52	29	47	14	
48	27	2	28	48	7	
49	76	51	27	49	15	
50	26	1	26	50	8	

TABLE 83. **RF I/O Connector Pinout**

The following table gives the pinouts for the SMB coaxial connectors used to connectors used to connect the A/D, D/A, and DDS signals. All connectors are 50 ohm type: inputs use 50 ohm termination to ground and outputs are sourced through 50 ohm resistors on the module PCB.

RF Omnibus Connector Reference Designator	Function
J1	A/D Channel 0 In
J2	A/D Channel 1 In
J3	D/A Channel 1 Out
J4	D/A Channel 0 Out
J5	External Clock In
J6	DDS Out
J7	DDS Auxiliary Channel Out

TABLE 84. **RF Coaxial I/O Connector Pinout**

Functions

Data In and Out

The logic contains two 32 bitx1024 deep FIFOs, one for each direction of data flow. The FIFOs are used to buffer data and as a transition from the A/D or D/A time domain to the Omnibus H1 time domain.

Data read by the Omnibus host from the A/D converter section uses the following format (depending on mode) on the 32-bit data bus.

	Sample Time 1	Sample Time 0
Bit Number:	31 - 16	15 - 0
Bit Field:	Channel 0 (12-bit sign extended)	Channel 0 (12-bit sign extended)

Mode 0: 1 Channel, 12-bit Resolution

	Sample Time 0	
Bit Number:	31 - 16	15 - 0
Bit Field:	Channel 1 (12-bit sign extended)	Channel 0 (12-bit sign extended)

Mode 1: 2 Channel, 12-bit Resolution

	Sample Time 3	Sample Time 2	Sample Time 1	Sample Time 0
Bit Number:	31 - 24	23 - 16	15 - 8	7 - 0
Bit Field:	Channel 0	Channel 0	Channel 0	Channel 0

Mode 2: 1 Channel, 8-bit Resolution

	Sample Time 1		Sample Time 0	
Bit Number:	31 - 24	23 - 16	15 - 8	7 - 0
Bit Field:	Channel 1	Channel 0	Channel 1	Channel 0

Mode 3: 2 Channel, 8-bit Resolution

Data written by the host to the D/A FIFOs use the identical formatting, with the exception that the most significant four bits of all 12-bit data fields in modes 0 and 1 are treated by the module as don't care bits.

Run Control Registers

This register toggles the run bits for the A/Ds and the D/As. The bits for this register default to "0". While the run is disabled, the A/D FIFOs are cleared. Run bit of "1" will cause the module to begin moving data.

Bit Number:	1	0
Bit Field:	DACRUN	ADRUN

FIGURE 63. **RF Run Control Register**

Bit Field Name	Function
ADRUN	A/D Run
DACRUN	D/A Run

TABLE 85. **RF Run Control Register Definition**

A/D Control Registers

This register controls the FIFO interrupt level. The user must choose one of bits 2..6. If an external run signal is used, bit 8 must be set to enable external run. If an output signal is needed, Sync out will drive out the A/D clock signal when Sync enable is set.

Bit Number:	31-10	9	8	7	6	5	4	3	2	1	0
Bit Field:	Reserved	SYNC	EXTRUN	Reserved	ADTQ	ADH	ADQ	ADF	ADE	Reserved	Reserved

FIGURE 64. RF A/D Control Register

Bit Field Name	Function
ADE	Interrupt enable - A/D FIFO empty.
ADF	Interrupt enable - A/D FIFO full.
ADQ	Interrupt enable - A/D FIFO quarter full.
ADH	Interrupt enable - A/D FIFO half full.
ADTQ	Interrupt enable - A/D FIFO three quarters full.
EXTRUN	External Run Enable
SYNC	Sync. Enable

TABLE 86. RF A/D Control Register Definition

D/A Control Register

The D/As can be placed in low power mode by setting the SLEEP bit. This register also controls the FIFO interrupt level. The user must choose one of bits 2..6. If an external run signal is used, bit 8 must be set to enable external run. If an output signal is needed, Sync out will drive out the D/A clock signal when Sync enable is set.

Bit Number:	31-11	10	9	8	7	6	5	4	3	2	1	0
Bit Field:	Reserved	SYNC	EXTRUN	DARESET	Reserved	DATQ	DAH	DAQ	DAF	DAE	Reserved	DACS

FIGURE 65. RF D/A Control Register

Bit Field Name	Function
SLEEP	D/A sleep mode enable, default = 0
DAE	Interrupt enable - D/A FIFO empty.
DAF	Interrupt enable - D/A FIFO full.
DAQ	Interrupt enable - D/A FIFO quarter full.
DAH	Interrupt enable - D/A FIFO half full.
DATQ	Interrupt enable - D/A FIFO three quarters full.
DARESET	D/A FIFO Reset.
EXTRUN	External Run Enable.
SYNC	Sync. Enable

TABLE 87. **RF D/A Control Register Definition**

D/A Run Control: Due to the delay imposed by the offset and gain correction pipeline, output of the top element(s) in the D/A FIFO memory will not occur until ten D/A sample clock cycle after the assertion of the D/A run function. Please note that an additional synchronization delay of up to one sample period may be imposed to the asynchronous run inputs (either the software or hardware versions) which is due to synchronization of the run signal to the D/A sample clock.

A/D and D/A Clock Configuration

The RF module includes numerous clock sources for driving the sample rate of the analog converters. Converter sample rates may be driven by the onboard DDS synthesizer device, by an external analog clock source received on SMB coaxial connector J5, by an external LVDS compatible digital clock source received via the OMNIBUS I/O connector, or by the OMNIBUS host’s DDS timebase.

The A/D and D/A clock configuration registers determine the source of the A/D and D/A sample clocks. One register exists for each I/O direction, allowing different clocks to be selected for the input data versus the output data. Only one of the bits in each register should be set at one time.

Bit Number:	31-4	3	2	1	0
Bit Field:	Reserved	SYNCIN	BBDDSC	EXTC	DDSC

FIGURE 66. **RF A/D & D/A Clock Configuration Registers**

Bit Field Name	Function
DDSC	Onboard DDS clock generator
EXTC	External clock via connector J5
BBDDSC	OMNIBUS host DDS clock
SYNCIN	External LVDS clock via I/O connector

TABLE 88. **RF A/D & D/A Clock Configuration Register Definition**

DDS Control Register

First, the DDS chip must be taken out of reset by writing a '0' to bit 0. To write data words to the DDS chip, IOMOD0 and address 0x800 **must** be written. This is done by writing to the DDS data address. Once the data has been written to the DDS chip, a '1' must be written to bit 1 of the DDS control register. This will cause the new settings for the DDS chip to be activated.

Bit Number:	0	1	2	3	4
Bit Field:	RESET	IO_UD	Reserved	FSK	SHAPE

FIGURE 67. **RF DDS Control Register**

Bit Field Name	Function
RESET	Reset, default = “1”
IO_UD	io_ud
FSK	fsk
SHAPE	shape

TABLE 89. **RF DDS Control Register Definition**

DDS Data Registers

The DDS registers are mapped one-for-one starting at IOMOD0 address 0X800. For more information on programming the DDS refer to the Analog Devices (AD9852) documentation.

Bits	Function
0..7	Data to the DDS chip.

TABLE 90. RF DDS Data Register

A/D and D/A Mode Register

The A/D and D/A mode register is shown in the table below. One of the bits (0..3) must be set.

Bit Number:	0	1	2	3
Bit Field:	Mode 0	Mode 1	Mode 2	Mode 3

FIGURE 68. RF A/D and D/A Mode Register

Bit Field Name	Function
Mode 0	1-Channel, 12-bit
Mode 1	2-Channel, 12-bit
Mode 2	1-Channel, 8-bit
Mode 3	2-Channel. 8-bit

TABLE 91. RF A/D and D/A Mode Register Definition

A/D and D/A Gain Control Registers

The A/D and D/A gain control registers contain the gain control information. The module will do a *Y*=*MX* + *B* operation of the data. The gain data represents the *M* coefficient. The range is 0x0000 to 0x1fff, with 0x1000 equal to a coefficient to 1.0.

A/D and D/A Offset Control Registers

The A/D and D/A offset control registers contain the offset control information. The module will do a *Y*= *MX* + *B* operation on the data. The offset data represents the *B* coefficient. The A/D offset coefficients are then signed (i.e. a value of 10 written to a coefficient register will add 10 counts to the offset position of the returned data, while -10 will subtract ten counts). The D/A offset data is unsigned and centered around hexadecimal value 0x800 (i.e. a value of 0x80a will add ten counts to the D/A output offset, while 0x7f6 will subtract ten counts).

ID ROM

The ID ROM data is serially input or output through the ID ROM bit 0. The data is clocked by the ID ROM serial clock (bit 1). When there is no more data to be written out, the data bit should be set.

Bit Number:	0	1
Bit Field:	SDATA	SCLOCK

FIGURE 69. RF ID ROM Register

Bit Field Name	Function
SDATA	Serial Data
SCLOCK	Serial Clock

TABLE 92. RF ID ROM Register Definition

EEPROM Register

The serial configuration EEPROM that configures the logic can be reprogrammed with the use of two registers. The EEPROM programming interface is enabled by writing a one bit to the programming enable register. The EEPROM programming register can be used to input ROM data. Once the EEPROM has been programmed bits 0 and 1 of the SROM programming enable register must be set to a '1'.

Bit Number:	0	1
Bit Field:	SDATA	SCLOCK

FIGURE 70. RF EEPROM Register

Bit Field Name	Function
SDATA	Serial Data
SCLOCK	Serial Clock

TABLE 93. **RF EEPROM Register Definition**

Bit	Function
0	Serial ROM program enable, default = 0

TABLE 94. **RF EEPROM Programming Enable Register**

Caution: The EEPROM holds the factory calibrated coefficients. Therefore, do not accidentally erase these factory calibrated coefficients.

Some Considerations About Using the RF

Performance Issues

The table below shows the maximum data rates achievable. The data rates for the ChicoPlus and Hombre are listed as Chico family baseboards. To get the data rates on all of the other DSP baseboards the rate must be divided by four, as shown in the table below.

Bit Field Name	Function
Mode 0	1-Channel, 12-bit @ 32 MHz (maximum) - Chico family baseboards. 1-Channel, 12-bit @ 8 MHz (maximum) - All other DSP baseboards.
Mode 1	2-Channel, 12-bit @ 16 MHz (maximum) - Chico family baseboards. 2-Channel, 12-bit @ 4 MHz (maximum) - All other DSP baseboards.
Mode 2	1-Channel, 8-bit @ 64 MHz (maximum) - Chico family baseboards. 1-Channel, 8-bit @ 16 MHz (maximum) - All other DSP baseboards.
Mode 3	2-Channel, 8-bit @ 32 MHz (maximum) - Chico family baseboards. 2-Channel, 8-bit @ 8 MHz (maximum) - All other DSP baseboards.

FIGURE 71. **Maximum Data Rates**

CHAPTER 13

SD Module

Module Introduction

The SD module gives the target processor card with four channels of high quality, professional grade 24-bit, 96 kHz sigma-delta analog A/D and D/A. Each channel has extremely low noise allowing over 100 dB of S/N ratio for ultra-clean audio signal acquisition and playback as a result of the sigma-delta architecture. The sigma-delta filter mechanism also defeats high frequency signals with a very effective digital filter in the A/D, preventing errors form out-of-band signals.

The SD module has balanced (differential) input capability for maximum noise rejection at the front end. Careful attention was paid to selecting the op-amps to minimize noise and distortion, providing a clean front-end to the A/D converter.

The SD module control logic conveniently maps the A/D and D/A interface as a parallel interface and the memory-mapped register set is easily accessed by the DSP. This relieves the DSP of the clumsy serial interface usually associated with the sigma-delta A/Ds, putting more valuable DSP cycles to work for data processing.

The module implements two Crystal CS5329 dual 24-bit A/D converters and two AKM AK4324 dual 24-bit D/A converters along with input and output circuitry and on-board power regulation. The module derives sample timing from the baseboard's AD9850 direct digital synthesizer for precise audio sample rates. The module can run at all current standard audio rates (32 kHz, 44.1 kHz, 48 kHz, and 96 kHz) and sends/receives full 24-bit resolution.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

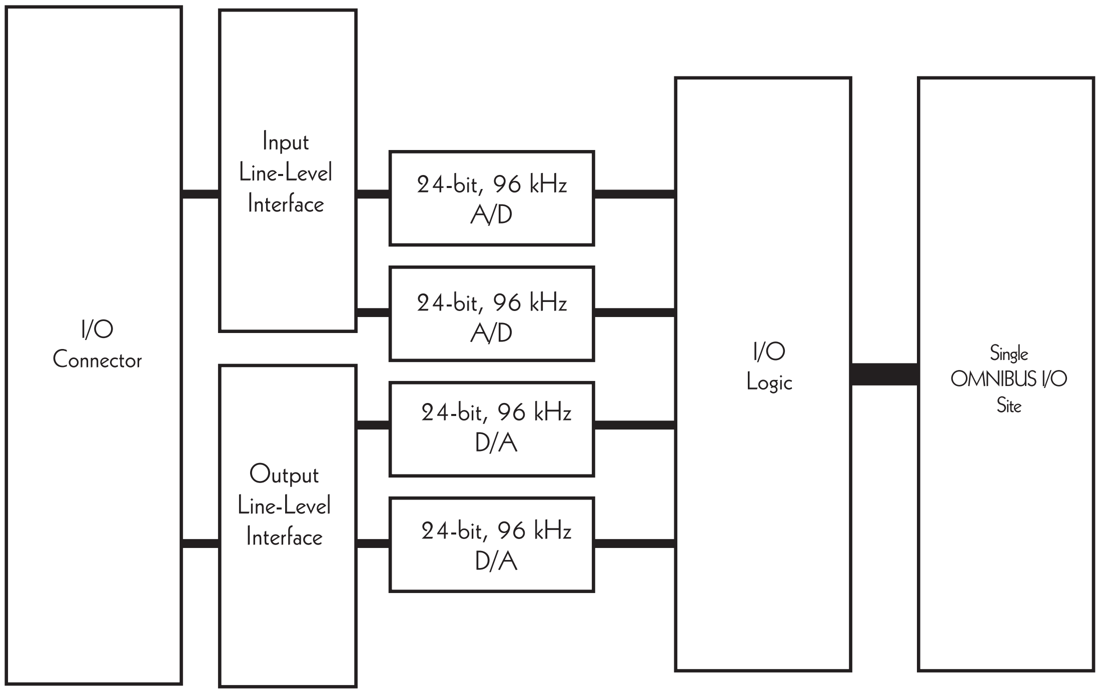


FIGURE 72. SD Block Diagram

Bus Type:	Compatible with all I.I. products. Consumes one interrupt to host. Wait-state depends on host platform.	**Digital Filter Characteristics:**	Passband = .4604 x sample rate Fs; Passband ripple = .005 dB; Stopband = .5542 x Fs min.; 63.42 X Fs max.
Power Requirements:	5 V @ 60 mA; +15 V @ 80 mA; -15 V @ 80 mA	**Interface to DSP:**	Memory-mapped registers using FPGA interface.
Physicals:	OMNIBUS mezzanine card; 2.0" X 4.6"	**Conversion Trigger Sources:**	Timers
A/D Converters:	Two stereo A/D chips; Crystal Semiconductor CS5396; Delta-sigma architecture for low noise, high resolution.	**D/C Converters:**	Two stereo output AKM AK4324; Delta-sigma architecture for low noise, high resolution.
Resolution:	24-bit	**Resolution:**	24-bit
Sample Rate (Fs):	2-96 kHz, programmable via host Timer.	**Oversampling:**	128 X
Over Sampling:	128 X	**Output Range:**	Line levels +/- 2 V unbalanced (single ended)
Analog Input Range:	Professional levels, 13 V rms	**Output Analog Filter:**	2 pole, 50 kHz typical.
S/N Ratio:	100 dB	**Update Rate:**	30-96 kHz, programmable via host Timer.
THD:	-100 dB	**Dynamic Range:**	105 dB
Dynamic Range:	100 dB	**S/N Ratio:**	100 dB
DC Gain Error:	5%	**THD + Noise:**	92 dB
Input Type:	Balanced or unbalanced (differential or single-ended)	**Output Control:**	Digital de-emphasis and attenuation control.
Input Impedance:	7 K	**Conversion Trigger Sources:**	Timers
Group Delay:	34/Fs S	**Interface to DSP:**	Memory-mapped registers using FPGA interface.

The target Peripheral Library provides support for each of the A4D2 functions in multiplexed and non-multiplexed, and the following sections give descriptions of the support routines. Refer to the **sdadc.h**, or **sddac.h** or files located in the `c:/<target board>/Include/Target/Analog` director and **Omnibus.hlp or Zuma.hlp** Windows helpfile for complete details on each function.

Function Name	**Operation**
SD_read_adc()	Reads A/D converter sample results.
SD_read_dac()	Reads D/A converter sample results.
SD_cal_adc()	Calibrates the SD A/Ds.
SD_clock()	Controls the A/Ds left/right clock.
SD_set_sample_freq()	Sets the sample frequency for the SD A/Ds and D/As.
SD_write_dac()	Writes data to D/A converters.
SD_mute_dac()	SD D/A mute control.
SD_set_dac_deemphasis()	SD D/A deempahsis control.
SD_set_dac_speed_range()	Sets the D/As clock mode.
SD_power_adc()	Sets the A/D power state
SD_power_dac()	Controls the SD D/As powerdown mode.
SD_read_idrom()	Read identification information from the idrom.
SD_write_idrom()	Writes identification information to the idrom.
SD_interrupt()	Sets which interrupt for the module to use.

TABLE 95. **C Language SD Functions**

The SD library functions can be divided into several groups:

1. Sample rate control.
2. A/D and D/A data I/O.
3. D/A mute and de-emphasis control.
4. Module power-down control.
5. Identification readback.

Sample Rate Control

The sample rate to the SD may be precisely controlled via the module site's onboard 9850 DDS timer. The **SD_sample()** function controls the sampling rate used by both the A/D and D/A converters within the application.

A/D and D/A Data I/O

Accessing the data registers for the A/D and D/A's on the SD modules is straightforward.

To read previously converted data from the A/Ds, use the **SD_read_adc()** function, which takes the channel number to read as an argument. Note that before using the A/D subsystem, the **SD_cal_adc()** function should be called to calibrate the A/D.

To write data to the D/As, use the **SD_write_dac()** function, which takes the channel number and the value to be output as arguments.

Identification Readback

The **SD_read_idrom()** function is used to read the identification ROM on the SD to check its identity and revision level. The function fills out an `SD_ID` structure with the information stored in the ID ROM on the module. The `SD_ID` structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string "SD").
2. Module revision level (single character revision level).
3. Checksum.

This data is preprogrammed at the factory and should not be altered by the user.

Memory Mapping

The SD module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** file included with the baseboard Development Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
D/A Channel 0 Load	IOMOD0 + 0	IOMOD4 + 0	IOMOD8 + 0
D/A Channel 1 Load	IOMOD0 + 1	IOMOD4 + 1	IOMOD8 + 1
D/A Channel 2 Load	IOMOD0 + 2	IOMOD4 + 2	IOMOD8 + 2
D/A Channel 3 Load	IOMOD0 + 3	IOMOD4 + 3	IOMOD8 + 3
D/A Clock Speed Range	IOMOD0 + 4	IOMOD4 + 4	IOMOD8 + 4
D/A De-emphasis Select	IOMOD0 + 5	IOMOD4 + 5	IOMOD8 + 5

Function	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
D/A Mute Control	IOMOD0 + 6	IOMOD4 + 6	IOMOD8 + 6
D/A Power Down Control	IOMOD0 + 7	IOMOD4 + 7	IOMOD8 + 7
A/D Channel 0 Read	IOMOD1 + 0	IOMOD5 + 0	IOMOD9 + 0
A/D Channel 1 Read	IOMOD1 + 1	IOMOD5 + 1	IOMOD9 + 1
A/D Channel 2 Read	IOMOD1 + 2	IOMOD5 + 2	IOMOD9 + 2
A/D Channel 3 Read	IOMOD1 + 3	IOMOD5 + 3	IOMOD9 + 3
A/D Power Down Control	IOMOD1 + 4	IOMOD5 + 4	IOMOD9 + 4
A/D Calibration Control	IOMOD1 + 5	IOMOD5 + 5	IOMOD9 + 5
Interrupt Control	IOMOD1 + 6	IOMOD5 + 6	IOMOD9 + 6
IDROM	IOMOD3	IOMOD7	IOMOD11

TABLE 96. **SD Memory Map**

Interrupt Usage

The SD uses a single I/O bus interrupt input to the baseboard processor to synchronize the CPU to the programmed sample rate timebase. This interrupt pulse may be used to trigger either CPU interrupts or DMA synchronization events, where applicable, in order to move digital audio data to/from the SD and the host processor's memory.

The SD's interrupt pin selection is programmable from the interrupt control address shown above. The module may be programmed to assert interrupts on either available interrupt input. At device power up or after reset, the module initializes with interrupt drive turned off, to avoid potential hardware conflicts with other modules or external hardware. The table below shows the available interrupt drive modes.

Interrupt Control Register Value	Interrupt Routing (OMNIBUS Slot 0)	Interrupt Routing (OMNIBUS Slot 1)	Interrupt Routing (OMNIBUS Slot 2)
0	Interrupts not driven to host.	Interrupts not driven to host.	Interrupts not driven to host.
1	Interrupt on external int 0.	Interrupt on external int 2.	Interrupt on external int 4.
2	Interrupt on external int 1.	Interrupt on external int 3.	Interrupt on external int 5.
3	Reserved	Reserved	Reserved

TABLE 97. **SD's Interrupt Drive Modes**

Pin Connector I/O

The SD output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* section for additional details on how to connect to the SD's I/O pins.

SD Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50 Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1..35	34..50, 83..100	9..25, 58..75	1..25, 41..50	1..35		Reserved
36	33	8	40	36	1	Audio Output Channel 0
37	82	57	39	37	9	GND
38	32	7	38	38	2	Audio Output Channel 1
39	81	56	37	39	10	GND
40	31	6	36	40	3	Audio Output Channel 2
41	80	55	35	41	11	GND
42	30	5	34	42	4	Audio Output Channel 3
43	79	54	33	43	12	Audio Input Channel 0B
44	29	4	32	44	5	Audio Input Channel 0A
45	78	53	31	45	13	Audio Input Channel 1B
46	28	3	30	46	6	Audio Input Channel 1A
47	77	52	29	47	14	Audio Input Channel 2B
48	27	2	28	48	7	Audio Input Channel 2A
49	76	51	27	49	15	Audio Input Channel 3B
50	26	1	26	50	8	Audio Input Channel 3A

TABLE 98. **SD I/O Connector Pinout**

Functions

Audio Inputs

The SD implements four channels of 24-bit analog to digital conversion via two stereo A/D converters (the Crystal CS5396). These serial converters are synchronously triggered by a single master clock, which is provided by the I/O bus host board's direct digital synthesizer channel. This master clock is then divided down internally by both the A/Ds and onboard logic to generate a serial bit clock and a left/right channel clock for the serial interface. A serial/parallel converter in onboard logic serves as an interface between the parallel I/O bus and the serial interface to the A/D devices.

The SD must take its master clock from the host's DDS output due to the high precision required in sampling high quality audio signals. The industry standard sample rates of 32, 44.1, 48, and 96 kHz may be precisely generated by the DDS. Please note that since the SD derives its master clock for all four A/D and all four D/A channels, both input sampling and output signal generation must be performed at the same sample rates (i.e. no multirate processing is possible on the SD module). Multiple host boards may be used to generate multirate audio processing systems based on the SD.

Line level balanced (differential) or unbalanced (single-ended) audio signals are presented to the SD via the IN0A/B, IN1A/B, IN2A/B, and IN3A/B pins on the external connectors. IN0A and IN0B are AC coupled through an input filter and level shifting interface to channel zero, IN1A and IN1B to channel two, etc. The following diagrams give the schematics for the input interface circuitry, which takes professional line level audio (18V peak to peak or 13Vrms) and presents a differential DC offset output appropriate to the CS5329.

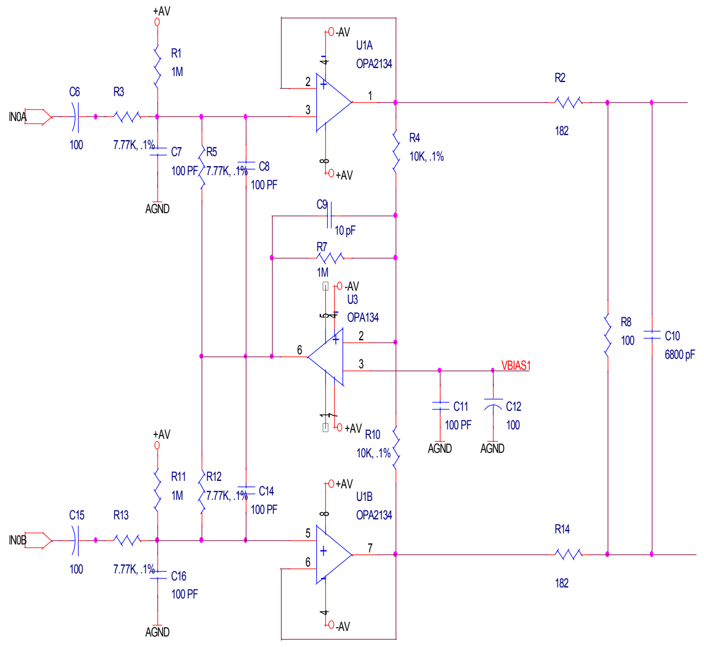


FIGURE 73. **SD Channel 0 Audio Input Circuitry**

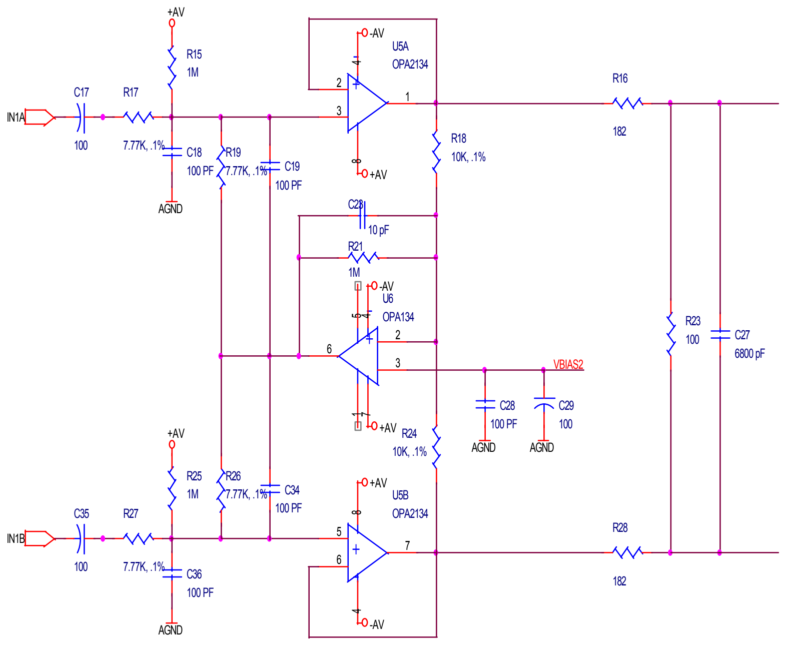


FIGURE 74. **SD Channel 1 Audio Input Circuitry**

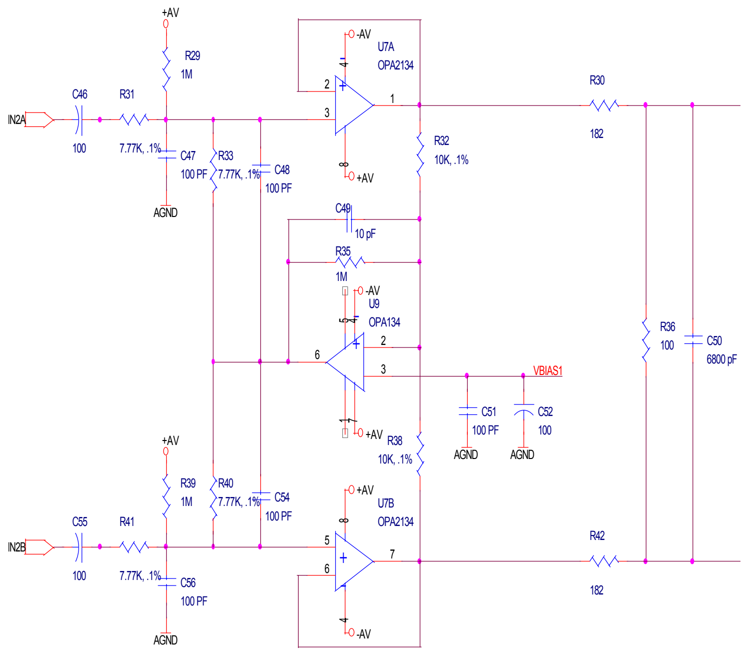


FIGURE 75. SD Channel 2 Audio Input Circuitry

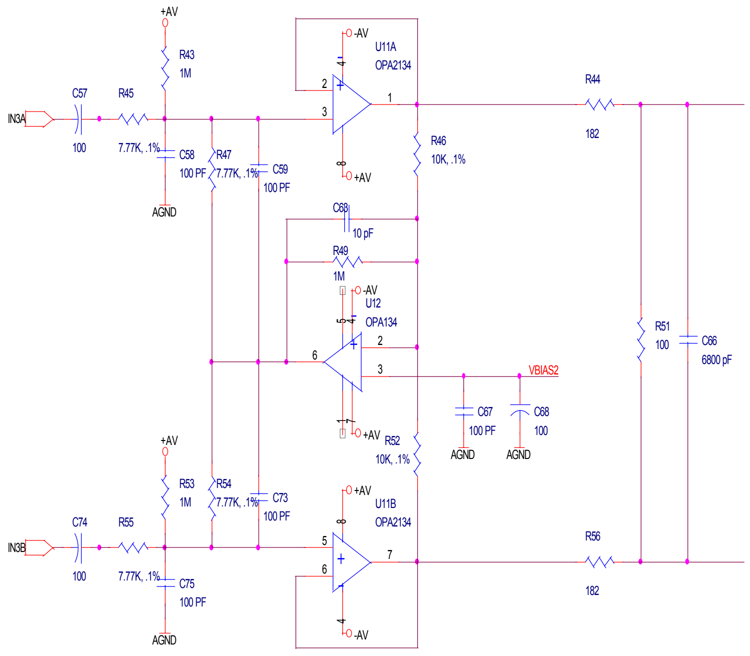


FIGURE 76. SD Channel 3 Audio Input Circuitry

The differential outputs of the level-shifting circuit are directly applied to the CS5329 converters (this circuit is repeated once for each input channel).

The following table gives a connection diagram showing how to connect the two types of input signals (balanced or unbalanced) to the differential inputs on each channel in order to obtain an inverted or non-inverted input.

Desired Signal Inversion	Balanced Input Signal	Unbalanced Input Signal
Noninverting	Positive side to A input Negative side to B input	Signal to A input Ground to B input
Inverting	Positive side to B input Negative side to A input	Signal to B input Ground to A input

Sample data returned by the A/D converters is in 24 bit two's complement format. The noninverting analog inputs result in return codes in the range of [8388607, -8388608]. Data on the upper eight bits of the read must be masked as these bits are not driven by hardware. Shifting up by 8 bits, then down by eight bits converts the masked 24-bit number to a sign-extended 32-bit number.

Audio Input Timing and Sample Rates

The A/D converters on the SD are controlled through a master clock, which through divide down circuitry, determines the final sample rate of all four of the input channels. To establish a particular sample frequency on the SD, it is necessary to calculate an appropriate master clock frequency and program the host board DDS timebase to generate the required frequency.

The master clock rate is always 256 times the required sample rate. Master clock frequencies for several standard audio sample rates are given in the table below. Other frequencies may be calculated simply by taking the required audio sample rate and multiplying by 256.

Audio Sample Rate (kHz)	Required Master Clock (DDS) Frequency (MHz)
32	8.1920
44.1	11.2896
48	12.2880
96	24.5760

TABLE 99. **Example SD A/D Audio Input Sample Rates with Master Clock Frequencies**

Sampling begins immediately following the DDS programming sequence. Both input and output are free-running once the timebase has been established. The A/D devices will convert continuously according to the sample rate determined by the master clock. The converters will continue to deliver data via the serial to parallel conversion hardware into the parallel holding registers regardless of whether or not the host CPU is retrieving data from the registers. There is no hardware provision for buffering or avoiding overflow conditions. Therefore, it is necessary to make sure that I/O bus host CPU software is responsive enough to read back all converted A/D data before the next conversion is complete.

Please note that it is not necessary to trigger conversion sequences in software, nor is there any supported means for triggering samples from external equipment.

Audio Input Powerdown Control

The SD provides a programmable powerdown feature for the A/Ds as a power saving measure. The A/D powerdown control register (see above for address) gives software control over the CS5329 power down pin. Placing the A/Ds in powerdown mode cuts power consumption from 900 mW/device to 1 mW/device. The following table gives register value information for controlling the powerdown state of the A/Ds (please note that only a single powerdown control is provided: all four A/D channels are controlled by the same signal, causing the entire input section of the SD to enter the low power state when software commands a powerdown).

A/D Powerdown Control Register Value	Mode
0	A/Ds in normal operational mode
1	A/Ds in powerdown mode

TABLE 100. SD A/D Powerdown Mode Control

After the hardware is reset or is powered-up, the SD hardware initializes with the A/D converters in powerdown mode. It is necessary to program the powerdown control register to zero and initiate a calibration before valid A/D conversion results will be returned. See the sections below on calibration and initialization issues for more information.

Audio Input Calibration

The SD module also provides direct software control over the calibration input pins of the CS5396 devices. Following a hardware reset, device power-up, or return from powerdown state, the A/Ds require an internally controlled calibration cycle to be performed before valid sample data will be returned by the devices. The A/D calibration is controlled by a CS5396 calibration request input pin, which in turn is driven by SD onboard logic and controlled by the A/D calibration control register. The following table gives the calibration control register values and their functions.

A/D Calibration Control Register Value	**Mode**
0	A/Ds in normal operational mode
1	A/Ds in calibration mode (rising edge triggers a new calibration)

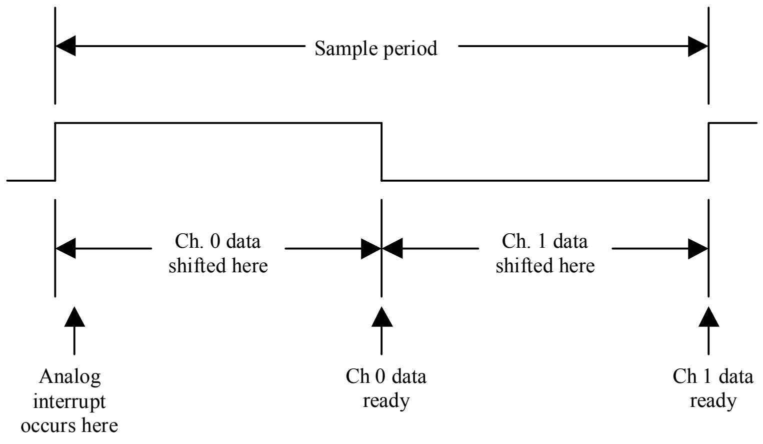
TABLE 101. **SD Calibration Control**

A/D calibrations may only be started after the device's voltage reference has stabilized. Reference stabilization typically requires approximately one half of a second, due to the large bypass and filter capacitances present on the A/D device's voltage reference output pins. A software delay following calibration of at least this long is necessary to assure maximum performance from the A/Ds.

Audio Data Read Timing

As discussed above, the SD module implements a single holding register for each A/D input channel, which contains the converter's digitized result of the input signal for each sample period. This register is updated immediately after the data is received from the corresponding A/D converter channel. Since the converters used on the SD module implement a serial data transmission scheme using a single transmission channel for each pair of A/D converter channels (i.e. one serial line for channels 0 and 1, and a second line for channels 2 and 3), these holding registers are updated at different times during the sample period depending on which channel is being received.

For example, in the case of A/D channels 0 and 1, channel 0's data is received first during each sample period, followed by data from channel 1. The first half of each sample period is used to serially receive the channel 0 data, which is then loaded into the channel 0 data holding register. The serial stream then switches to transmitting channel 1 data, which is received across the second half of each sample period and loaded into the channel 1 data holding register at the end of each sample period. The following diagram shows the timing relationships.



Interrupt signal generation occurs 3 OMNIBUS clock cycles after the beginning of each sample period, and notifies the host OMNIBUS board that data from the previous sample period is available in the holding registers.

This timing has an impact on data communications with the OMNIBUS host because host software needs to be aware of the mid-cycle updates occurring on the holding registers. OMNIBUS host read cycles should not occur near the holding register update points or incorrect data may result from the read operation. The easiest way to guarantee that reads occur correctly is to place the hardware accesses as early as possible in the interrupt handler code, or use DMA operations to move data from the SD module. These restrictions do not apply to the streaming engine products (ChicoPlus and Hombre) because hardware support is responsible for data movement to and from the OMNIBUS sites and this support is not in general programmable by the user.

Audio Output

The SD also implements four channels of 24-bit digital to analog conversion via two stereo D/A converters (the AKM AK4329). Similar to the A/D converters, these serial converters are synchronously triggered by a single master clock, which is provided by the I/O bus host board's direct digital synthesizer channel. This master clock is then divided down internally by both the D/As and onboard logic to generate a serial bit clock and a left/right channel clock for the serial interface. A serial/parallel converter in onboard logic serves as an interface between the parallel I/O bus and the serial interface to the D/A devices.

Line level audio signals are generated by the SD from the D/A device's differential outputs. The DC offsets are subtracted and the resultant AC signals filtered and summed to generate the line level outputs. The output circuit schematic is shown below (the circuit is repeated four times, once for each output).

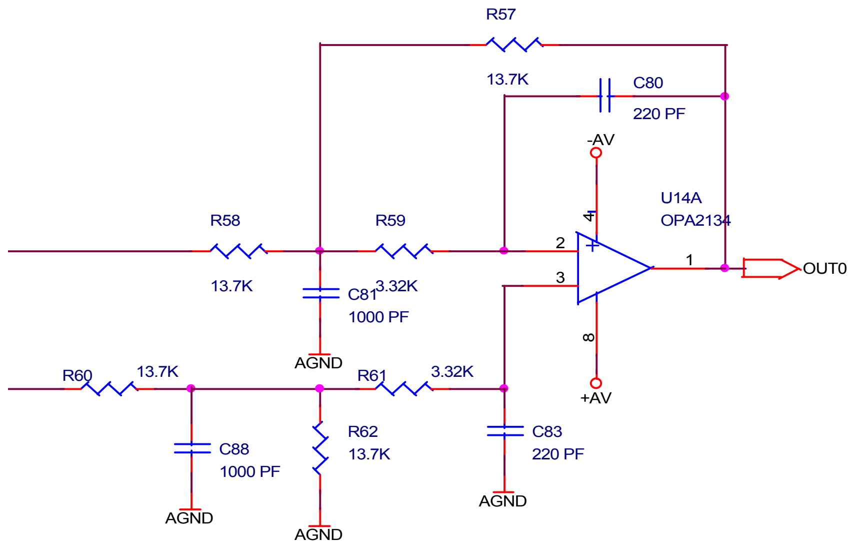


FIGURE 77. SD Channel 0 Audio Output Circuitry

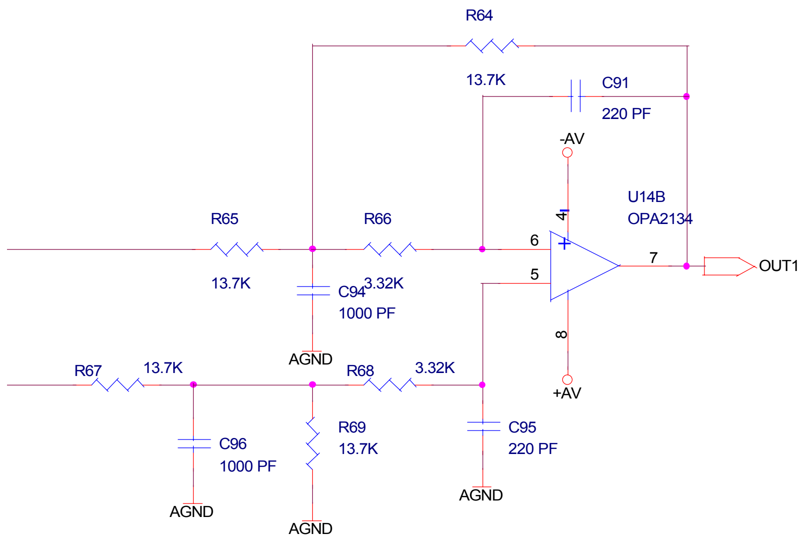


FIGURE 78. SD Channel 1 Audio Output Circuitry

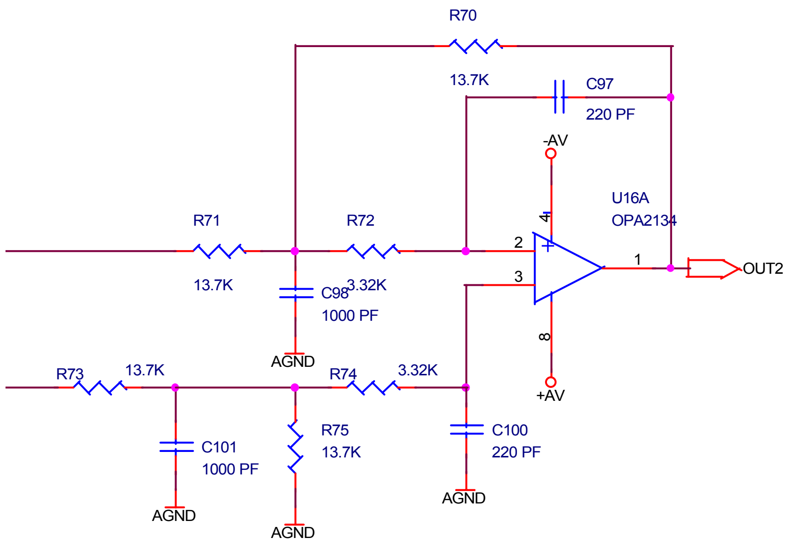


FIGURE 79. **SD Channel 2 Audio Output Circuitry**

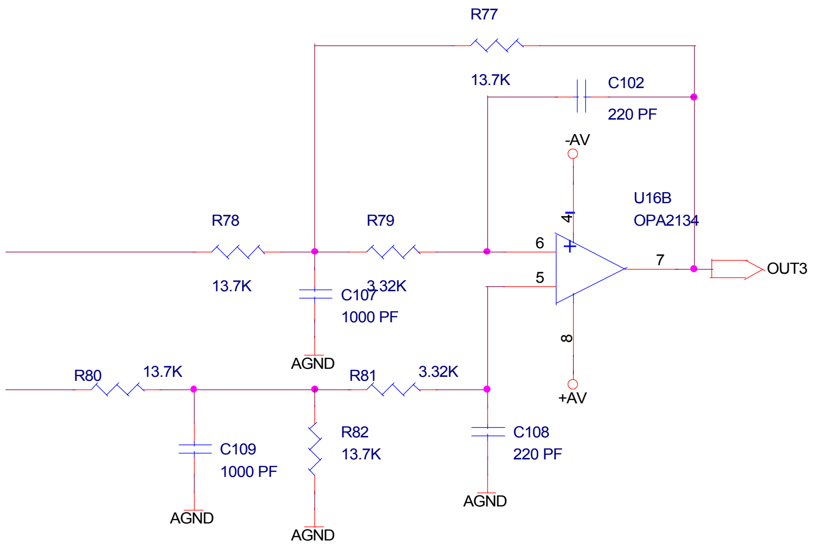


FIGURE 80. **SD Channel 3 Audio Output Circuitry**

The D/A converter interface expects input digital data in the form of two's complement numbers in the range of [8388607, -8388608]. The interface is compatible with direct host processor writes of 32-bit signed integers.

Audio Output Timing and Sample Rates

Industry standard audio sampling rates are also supported by the output circuitry of the SD. Similar to the input electronics, the outputs are driven by a master clock, which is nominally 256 times the frequency of the required sample rate. When running at sample rates faster than approximately 50 kHz, however, the AK4329 requires its input master clock rate to be divided by two, to 128 times the sample frequency. The SD onboard logic supports this requirement by employing a programmable divide down register which must be set by the host CPU when running at analog sample clock rates faster than 50 kHz.

The D/A clock speed range control register gives the host CPU control over the programmable division down of the D/A's master clock. Setting this register controls the frequency range within which the D/A will reliably convert. The following table gives example standard audio sample rate frequencies along with accompanying D/A clock range control register settings.

Audio Sample Rate (kHz)	Required Master Clock (DDS) Frequency (MHz)	Actual D/A Master Clock Frequency (MHz)	D/A Clock Speed Range Control Register Value
32	8.1920	8.1920	1
44.1	11.2896	11.2896	1
48	12.2880	12.2880	1
96	24.5760	12.2880	0

TABLE 102. **Example SD D/A Audio Sample Rates with Master Clock Frequencies and D/A Clock Speed Control Values**

Note: the AK4324 converter devices require that software activate the powerdown mode whenever a clock is not supplied to the parts. Always place the SD module D/A converters in power down mode first before deactivating or changing the frequency of the DDS clock.

Audio Output Muting and Digital De-emphasis

The audio output circuitry also supports the programmable muting and digital de-emphasis features of the AK4329. The D/A device's muting feature is controlled by the D/A muting control register, which has two bits for independent control of the two audio output pairs. The following table gives the register value combinations for the muting control register.

Register Value	Function
0	All D/A channels unmuted
1	D/A channels 0 and 1 muted
2	D/A channels 2 and 3 muted
3	All D/A channels muted

TABLE 103. **SD D/A Muting Control Register Values**

The D/A muting functions on all channels are activated on hardware reset or after powerup in order to mute any noise pulses caused by software startup on the host. After a hardware reset and before output signals can be generated by the D/A's, I/O bus host software must deactivate muting on the required channels.

Digital de-emphasis provides software programmable filtering for standard audio sample rates. The AK4329 provides a digitally controllable de-emphasis filter, which may be set to different passband frequencies dependent on the sample rate. The SD's D/A de-emphasis control register allows a global de-emphasis setting which controls the filters in all four D/A outputs simultaneously (a single de-emphasis control is congruent with the single sample rate limitation of the SD, as noted in the audio input section above). The following table gives the de-emphasis control register values appropriate to various sample rates.

Audio Sample Rate (kHz)	De-emphasis Control Register Value
32	3
44.1	0
48	2
96	2

TABLE 104. **SD D/A De-emphasis Control Register Values**

In general, the de-emphasis control register value should be set to match the required sample. Setting the control register to a value of one will deactivate the filter. Please note that the value of two for the 96kHz and 48 kHz rates is correct: the filter differentiates between the two settings by also looking at the clock rate control input, which would be set to one for the 96 kHz rate and zero for the 48 kHz rate.

Audio Output Powerdown Control

The SD also provides a powerdown feature for the D/A converters, which allows their power requirements to be cut significantly. The D/A powerdown control register uses a single control bit to activate the powerdown state of both D/A devices simultaneously. The following table gives the values for the powerdown control register.

Register Value	Function
0	All D/A devices in powerdown mode
1	All D/A devices in normal mode

TABLE 105. **SD D/A Powerdown Control Register**

Following system powerup or hardware reset, the D/As are placed in powerdown mode. Powerdown mode must be disabled by a write of one to the powerdown control register before D/A outputs will become active.

Audio Write Data Timing

Please see the section entitled Audio Read Data Timing for information on restrictions on the exact timing of data accesses between the OMNIBUS host and the SD module. The same restrictions apply to output as well as input, and in general data transmission should be performed as early as possible in the sample period (i.e. as soon as possible after reception of the analog interrupt from the SD module).

Initialization Issues

Please note the following SD initialization requirements. These apply after either a system powerup or a hardware reset is applied to the I/O bus host board.

1. Power down control after reset/powerup - all A/D and D/A devices come up in powerdown mode. The A/D and D/A powerdown control registers must be written to shift the conversion devices to normal operational mode before sampling and signal output can start.

2. Calibration - following deassertion of the powerdown feature of the A/D devices, a calibration command is necessary before valid sample data can be collected from the inputs.

3. D/A clock rate setup - the D/A's clock rate range must be selected via the D/A clock speed range control register for the required sample rate.

4. D/A muting active - D/A mute mode must be deactivated via the D/A muting control register before output signals can be generated.

CHAPTER 14

SD16 Module

Module Introduction

The SD16 module gives the target processor card with sixteen channels of high quality, professional grade 18-bit, 48 kHz sigma-delta analog A/D and D/A. Each channel has very low noise for ultra-clean audio signal acquisition and playback as a result of the sigma-delta architecture. In addition, each input channel employs delta sigma modulation with 64x over sampling, which provides an excellent anti-alias filter always automatically set at 1/2 the data rate. Careful attention was paid to selecting the op-amps to minimize noise and distortion, providing a clean front-end to the A/D converter.

The SD16 module control logic conveniently maps the A/D and D/A interface as a parallel interface and the memory-mapped register set is easily accessed by the DSP. This relieves the DSP of the clumsy serial interface usually associated with the sigma-delta A/Ds, putting more valuable DSP cycles to work for data processing.

The SD16 module implements eight PCM3001E dual 18-bit A/D and D/A converters along with line-level input and output circuitry and on-board power regulation. The module derives sample timing from the baseboard's AD9850 direct digital synthesizer for precise audio sample rates. The module sample and outputs rates are fully programmable in the range from 3 kHz to 48 kHz, which includes popular standards for audio rates (32 kHz, 44.1 kHz and 48 kHz) and sends/receives full 18-bit resolution.

The following block diagrams has been provided to show the conceptual arrangement of the component circuitry featured of the board. Following the block diagrams is the module specifications. These module specifications and block diagram should be referenced to while reviewing this module's chapter.

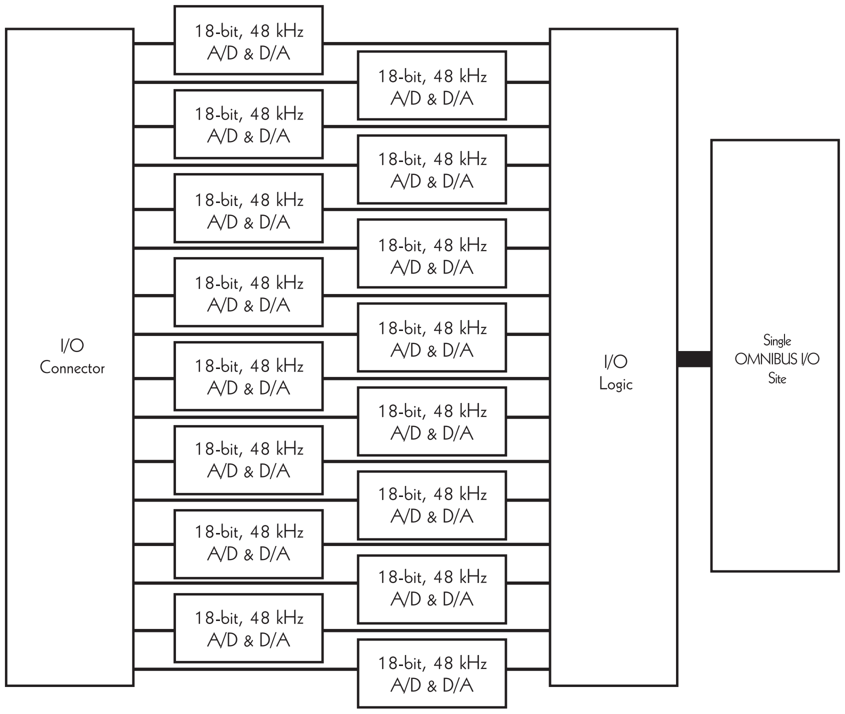


FIGURE 81. **SD16 Block Diagram**

Bus Type:	Compatible with all I.I. products; 18-bit. Consumes one interrupt to host. Wait-state depends on host platform.	**Digital Filter Characteristics:**	Passband = .454 x sample rate Fs; Passband ripple = .05 dB; Stopband = .583 x Fs min.
Power Requirements:	5 V @ 150 mA; +15 V @ 120 mA; -15V @ 120 mA	**Interface to DSP:**	Memory-mapped registers using FPGA interface.
Physicals:	OMNIBUS mezzanine card; 2.0” X 4.6”	**Conversion Trigger Sources:**	Timers
A/D Converters: (8 Stereo Codec Chips)	16 channels total; Burr-Brown PCM3001; Delta-sigma architecture for low noise, high resolution.	**D/C Converters: (8 Stereo Codec Chips)**	16 channels total; Burr-Brown PCM3001; Delta-sigma architecture for low noise, high resolution.
Resolution:	18-bit	**Resolution:**	18-bit
Sample Rate (Fs):	32-48 kHz, programmable via host Board Timer.	**Oversampling:**	64 X
Over Sampling:	64 X	**Output Range:**	+/- 10 V custom with resistor change.
Analog Input Range:	+/- 10 V	**Output Analog Filter:**	1 pole; -3 dB at 170 kHz
S/N Ratio:	92 dB	**Update Rate:**	32-48 kHz, programmable via host Board Timer
THD:	-88 dB	**Dynamic Range:**	97 dB
Dynamic Range:	92 dB	**S/N Ratio:**	96 dB
DC Gain Accuracy:	2%	**THD + Noise:**	-90 dB
Input Type:	Unbalanced (single-ended - AC coupled)	**Conversion Trigger Sources:**	Timers
Input Impedance:	10 K - AC coupled	**Interface to DSP**	Memory-mapped registers using FPGA interface
Group Delay:	17.4/Fs S		

The target Peripheral Library provides support for each of the A4D2 functions in multiplexed and non-multiplexed, and the following sections give descriptions of the support routines. Refer to the **sd16adc.h**, or **sd16dac.h** or files located in the **`c:/<target board>/Include/Target/Analog`** director and **Omnibus.hlp or Zuma.hlp** Windows helpfile for complete details on each function.

Function Name	**Operation**
SD16_initialize()	Resets the SD16 module.
SD16_control()	Set the control register on the module.
SD16_Interrupt()	Set up the interrupt control.
SD16_read_adc()	Reads A/D converter sample results.
SD16_write_dac()	Writes data to D/A converters.
SD16_read_dac()	Returns last data written to D/A.
SD16_read_idrom()	Read identification information.
SD16_write_idrom()	Write identification information.
SD16_correct_adc()	Adjusts the adc value to have the proper range.
SD16_correct_dac()	Adjusts the dac value to have the proper range.

TABLE 106. C Language SD16 Functions

The SD16 library functions can be divided into several groups:

1. Sample rate control
2. A/D and D/A data I/O
3. Identification readback

This data is preprogrammed at the factory and should not be altered by the user.

Sample Rate Control

The sample rate to the SD16 may be precisely controlled via the baseboard's 9850 DDS timer. The **timebase()** function controls the sampling rate used by both the A/D and D/A converters within the application.

A/D and D/A Data I/O

Accessing the data registers for the SD16 modules A/D and D/A's is straightforward.

To read previously converted data from the A/Ds, use the **SD16_read_adc()** function, which takes the channel number to read as an argument. Note that before using the A/D subsystem, the **SD16_initialize()** function should be called to calibrate the A/Ds.

To write data to the D/As, use the **SD16_write_dac()** function, which takes the channel number and the value to be output as arguments.

Identification Readback

The **SD16_read_idrom()** function is used to read the identification ROM on the SD16 to check its identity and revision level. The function fills out an `SD16_ID` structure with the information stored in the ID ROM on the module. The `SD16_ID` structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string “SD16”).
2. Module revision level (single character revision level).
3. Checksum.

This data is preprogrammed at the factory and should not be altered by the user.

Memory Mapping

The SD16 module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** file included with the baseboard Developer’s Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	**OMNIBUS Slot 0 Address**	**OMNIBUS Slot 1 Address**	**OMNIBUS Slot 2 Address**
D/A Channel 0 Load	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0
D/A Channel 1 Load	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1
D/A Channel 2 Load	IOMOD0 + 0x2	IOMOD4 + 0x2	IOMOD8 + 0x2
D/A Channel 3 Load	IOMOD0 + 0x3	IOMOD4 + 0x3	IOMOD8 + 0x3
D/A Channel 4 Load	IOMOD0 + 0x4	IOMOD4 + 0x4	IOMOD8 + 0x4
D/A Channel 5 Load	IOMOD0 + 0x5	IOMOD4 + 0x5	IOMOD8 + 0x5
D/A Channel 6 Load	IOMOD0 + 0x6	IOMOD4 + 0x6	IOMOD8 + 0x6
D/A Channel 7 Load	IOMOD0 + 0x7	IOMOD4 + 0x7	IOMOD8 + 0x7
D/A Channel 8 Load	IOMOD0 + 0x8	IOMOD4 + 0x8	IOMOD8 + 0x8

Function	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
D/A Channel 9 Load	IOMOD0 + 0x9	IOMOD4 + 0x9	IOMOD8 + 0x9
D/A Channel 10 Load	IOMOD0 + 0xA	IOMOD4 + 0xA	IOMOD8 + 0xA
D/A Channel 11 Load	IOMOD0 + 0xB	IOMOD4 + 0xB	IOMOD8 + 0xB
D/A Channel 12 Load	IOMOD0 + 0xC	IOMOD4 + 0xC	IOMOD8 + 0xC
D/A Channel 13 Load	IOMOD0 + 0xD	IOMOD4 + 0xD	IOMOD8 + 0xD
D/A Channel 14 Load	IOMOD0 + 0xE	IOMOD4 + 0xE	IOMOD8 + 0xE
D/A Channel 15 Load	IOMOD0 + 0xF	IOMOD4 + 0xF	IOMOD8 + 0xF
Codec Reset/Clock Control	IOMOD0 + 0x14	IOMOD4 + 0x14	IOMOD8 + 0x14
A/D Channel 0 Read	IOMOD1 + 0x0	IOMOD5 + 0x0	IOMOD9 + 0x0
A/D Channel 1 Read	IOMOD1 + 0x1	IOMOD5 + 0x1	IOMOD9 + 0x1
A/D Channel 2 Read	IOMOD1 + 0x2	IOMOD5 + 0x2	IOMOD9 + 0x2
A/D Channel 3 Read	IOMOD1 + 0x3	IOMOD5 + 0x3	IOMOD9 + 0x3
A/D Channel 4 Read	IOMOD1 + 0x4	IOMOD5 + 0x4	IOMOD9 + 0x4
A/D Channel 5 Read	IOMOD1 + 0x5	IOMOD5 + 0x5	IOMOD9 + 0x5
A/D Channel 6 Read	IOMOD1 + 0x6	IOMOD5 + 0x6	IOMOD9 + 0x6
A/D Channel 7 Read	IOMOD1 + 0x7	IOMOD5 + 0x7	IOMOD9 + 0x7
A/D Channel 8 Read	IOMOD1 + 0x8	IOMOD5 + 0x8	IOMOD9 + 0x8
A/D Channel 9 Read	IOMOD1 + 0x9	IOMOD5 + 0x9	IOMOD9 + 0x9
A/D Channel 10 Read	IOMOD1 + 0xA	IOMOD5 + 0xA	IOMOD9 + 0xA
A/D Channel 11 Read	IOMOD1 + 0xB	IOMOD5 + 0xB	IOMOD9 + 0xB
A/D Channel 12 Read	IOMOD1 + 0xC	IOMOD5 + 0xC	IOMOD9 + 0xC
A/D Channel 13 Read	IOMOD1 + 0xD	IOMOD5 + 0xD	IOMOD9 + 0xD
A/D Channel 14 Read	IOMOD1 + 0xE	IOMOD5 + 0xE	IOMOD9 + 0xE
A/D Channel 15 Read	IOMOD1 + 0xF	IOMOD5 + 0xF	IOMOD9 + 0xF
Interrupt Control	IOMOD1 + 0x16	IOMOD5 + 0x16	IOMOD9 + 0x16
IDROM	IOMOD3	IOMOD7	IOMOD11

TABLE 107. SD16 Memory Mapping

Interrupt Usage

The SD16 uses a single I/O bus interrupt input to the baseboard processor to synchronize the CPU to the programmed sample rate timebase. This interrupt pulse may be used to trigger either CPU interrupts or DMA synchronization events, where applicable, in order to move digital audio data to/from the SD16 and the host processor’s memory.

The SD16’s interrupt pin selection is programmable from the interrupt control address shown above. The module may be programmed to assert interrupts to the OMNIBUS host processor on either available OMNIBUS interrupt input. At device power up or after reset the module initializes with interrupt drive turned off, to avoid potential hardware conflicts with other modules or external hardware. The table below shows the available interrupt drive modes.

Interrupt Control Register Value	Interrupt Routing (I/O Module Slot 0)	Interrupt Routing (I/O Module Slot 1)
0	Interrupts not driven to host.	Interrupts not driven to host.
1	Interrupt on external int 0.	Interrupt on external int 2.
2	Interrupt on external int 1.	Interrupt on external int 3.
3	Reserved	Reserved

TABLE 108. SD16 Interrupt Routing

Pin Connector I/O

The SD16 output connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* section for additional details on how to connect to the SD16's I/O pins.

SD16 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50 Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1	100	75	25	1		AGND
2	50	25	24	2		LRCIN (CODEC sample clock)
3	99	74	23	3		AGND
4	49	24	22	4		DGND
5	98	73	21	5		AGND
6	48	23	20	6		Reserved
7	97	72	19	7		AGND
8	47	22	18	8		Reserved
9	96	71	17	9		Reserved
10	46	21	16	10		Reserved
11	95	70	15	11		Reserved
12	45	20	14	12		Reserved
13	94	69	13	13		Reserved
14	44	19	12	14		Reserved
15	93	68	11	15		Audio Input Channel 6
16	43	18	10	16		Audio Input Channel 7
17	92	67	9	17		Audio Input Channel 8
18	42	17	8	18		Audio Input Channel 9
19	91	66	7	19		Audio Input Channel 10
20	41	16	6	20		Audio Input Channel 11
21	90	65	5	21		Audio Input Channel 12
22	40	15	4	22		Audio Input Channel 13
23	89	64	3	23		Audio Input Channel 14
24	39	14	2	24		Audio Input Channel 15
25	88	63	1	25		Audio Output Channel 6
26	38	13	50	26		Audio Output Channel 7

SD16 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50 Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
27	87	62	49	27		Audio Output Channel 8
28	37	12	48	28		Audio Output Channel 9
29	86	61	47	29		Audio Output Channel 10
30	36	11	46	30		Audio Output Channel 11
31	85	60	45	31		Audio Output Channel 12
32	35	10	44	32		Audio Output Channel 13
33	84	59	43	33		Audio Output Channel 14
34	34	9	42	34		Audio Output Channel 15
35	83	58	41	35		Reserved
36	33	8	40	36	1	Reserved
37	82	57	39	37	9	Audio Input Channel 4
38	32	7	38	38	2	Audio Input Channel 5
39	81	56	37	39	10	Audio Output Channel 4
40	31	6	36	40	3	Audio Output Channel 5
41	80	55	35	41	11	Audio Output Channel 2
42	30	5	34	42	4	Audio Output Channel 3
43	79	54	33	43	12	Audio Output Channel 0
44	29	4	32	44	5	Audio Output Channel 1
45	78	53	31	45	13	Audio Input Channel 2
46	28	3	30	46	6	Audio Input Channel 3
47	77	52	29	47	14	Audio Input Channel 0
48	27	2	28	48	7	Audio Input Channel 1
49	76	51	27	49	15	AGND
50	26	1	26	50	8	AGND

TABLE 109. SD16 I/O Connector Pinout

Functions

Audio Inputs

The SD16 implements sixteen channels of 18-bit analog to digital conversion via eight stereo A/D converters. These serial converters are synchronously triggered by a single master clock provided by the OMNIBUS host board's direct digital synthesizer channel. This master clock is divided down internally by the onboard logic to generate a serial bit clock and a left/right channel clock for the serial interface to the codes. A serial/parallel converter in onboard logic serves as an interface between the parallel I/O bus and the serial interface to the A/D devices.

The SD16 must take its master clock from the host's DDS output due to the high precision required in sampling high quality audio signals. The industry standard sample rates of 32, 44.1 and 48 kHz may be precisely generated by the DDS. Please note that since the SD16 derives it master clock for all sixteen A/D and all sixteen D/A channels from the same clock source. Both input sampling and output signal generation must be performed at the same sample rates (i.e. no multi-rate acquisition is possible on the SD16 module). Multiple host boards may be used to generate multi-rate audio processing systems based on the SD16.

Single-ended audio signals with a maximum voltage range of +-10V are presented to the SD16 via the audio input pins on the I/O connector. Each signal is passed through an input amplifier for filtering and level matching before being presented to the corresponding A/D converter input. The following diagrams give the schematics for the input interface circuitry.

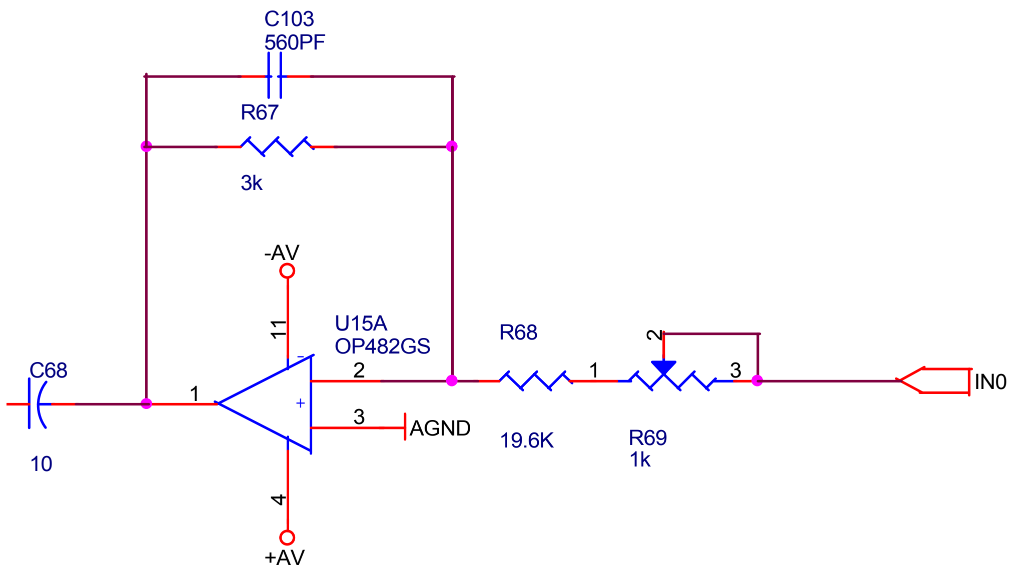


FIGURE 82. **SD16 A/D Channel 0 Input Amplifier**

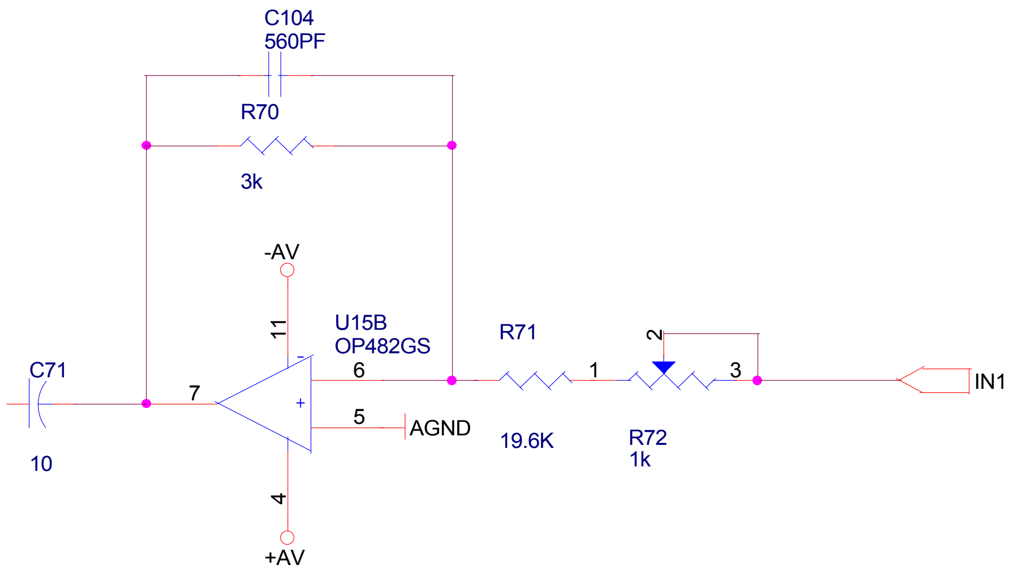


FIGURE 83. **SD16 A/D Channel 1 Input Amplifier**

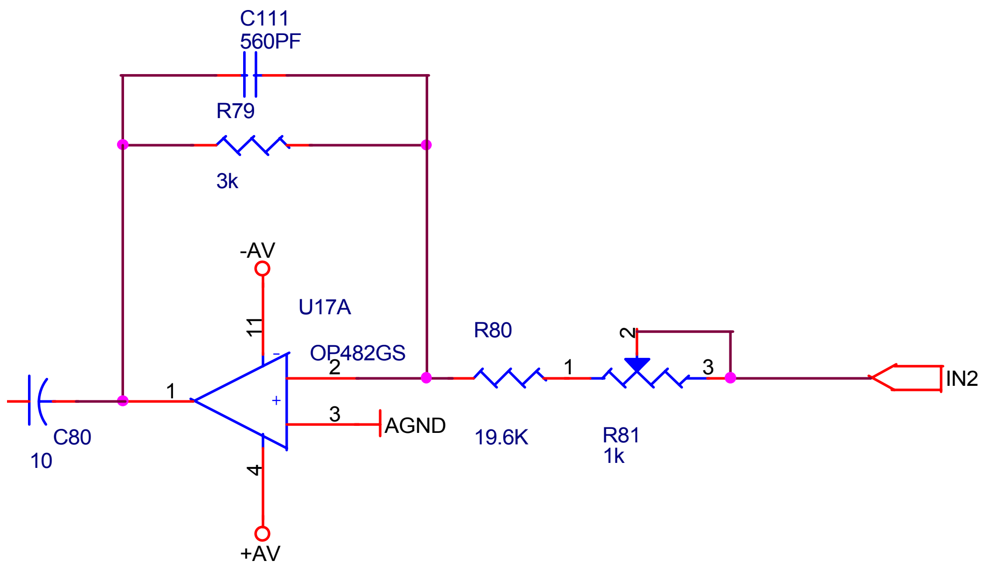


FIGURE 84. **SD16 A/D Channel 2 Input Amplifier**

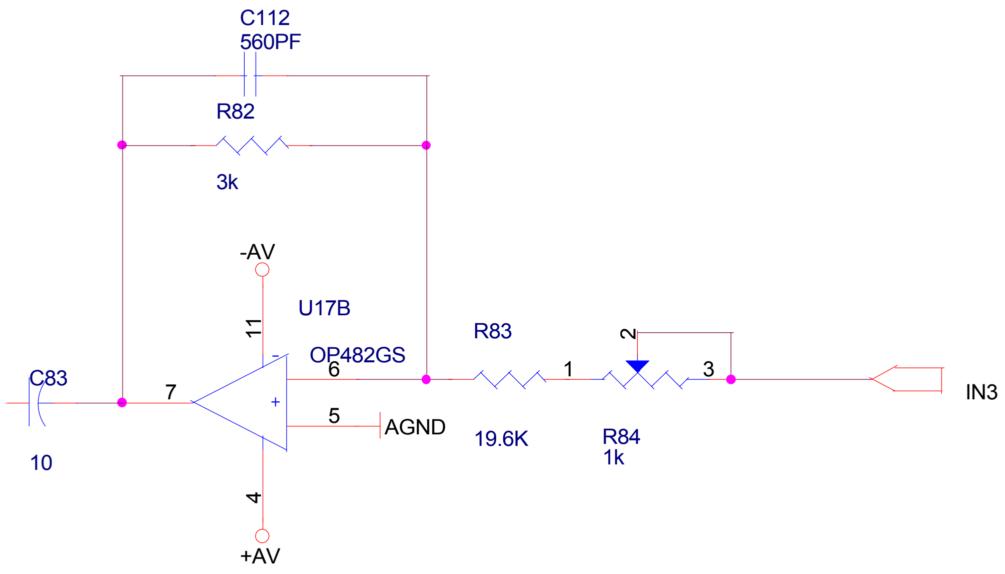


FIGURE 85. **SD16 A/D Channel 3 Input Amplifier**

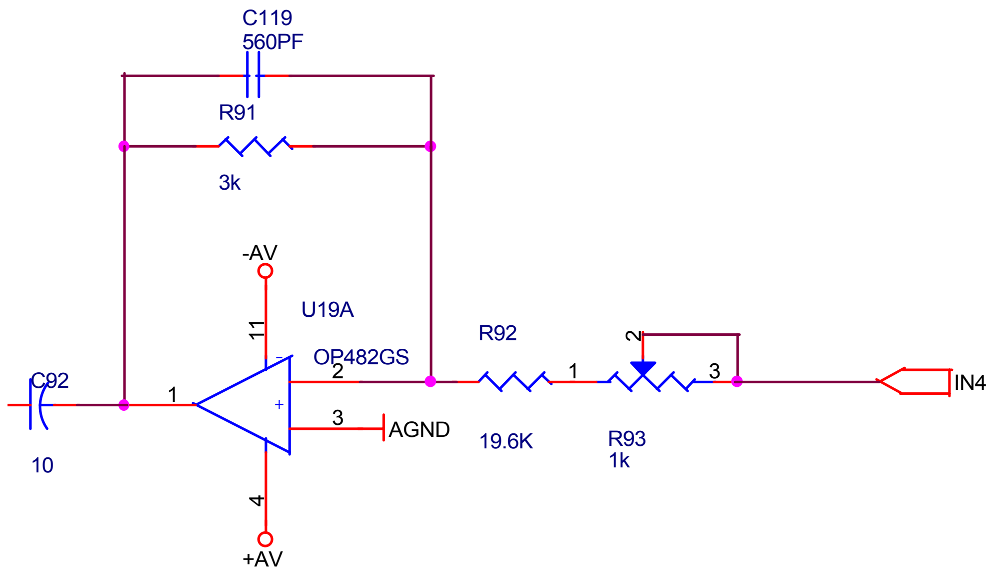


FIGURE 86. SD16 A/D Channel 4 Input Amplifier

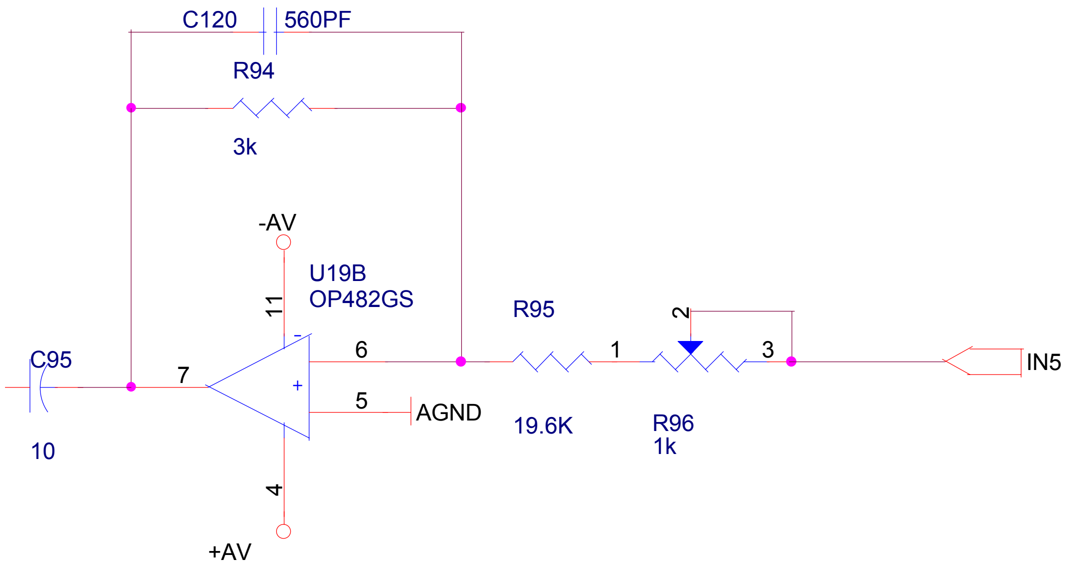


FIGURE 87. SD16 A/D Channel 5 Input Amplifier

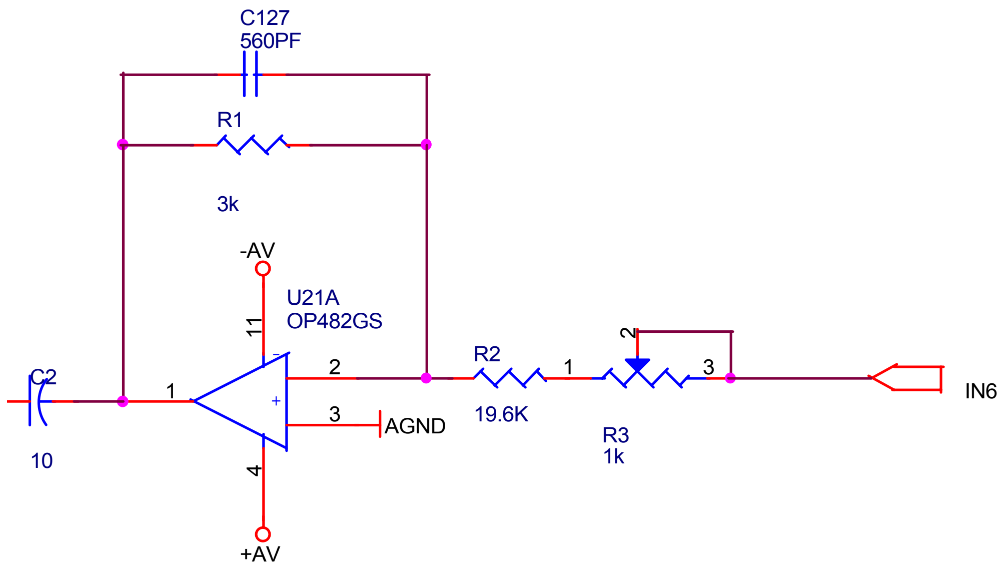


FIGURE 88. SD16 A/D Channel 6 Input Amplifier

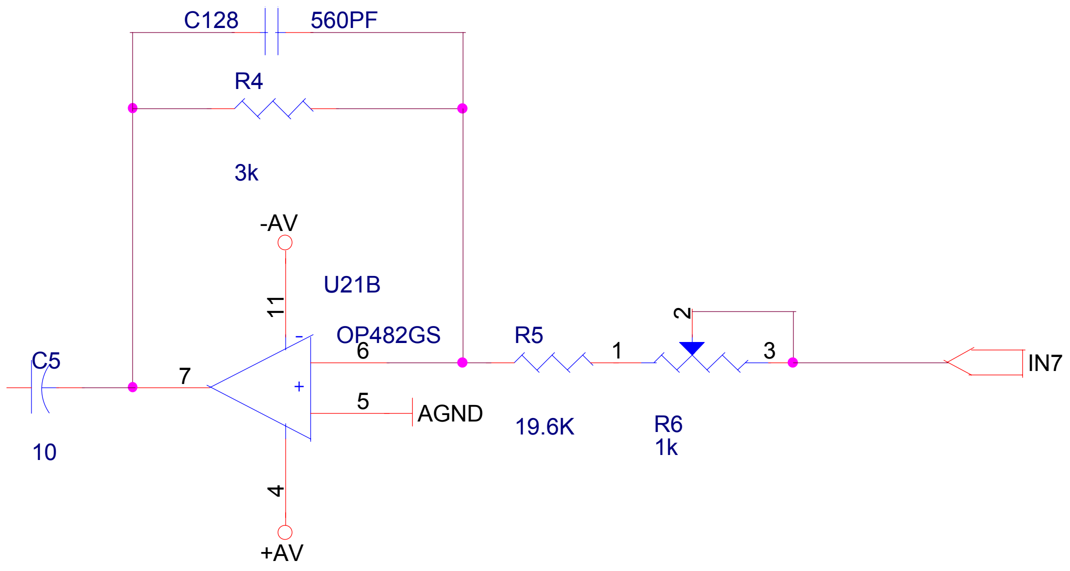


FIGURE 89. SD16 A/D Channel 7 Input Amplifier

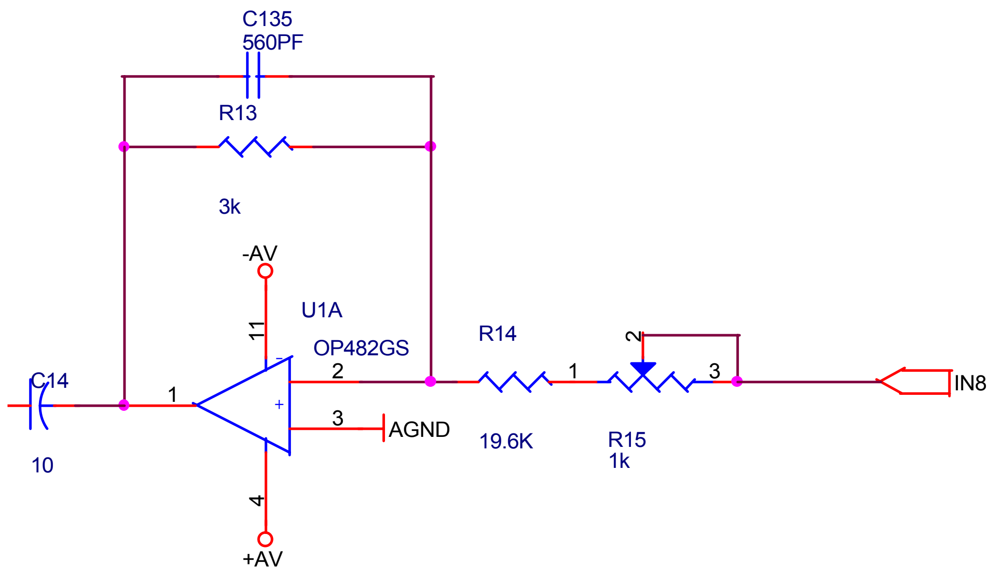


FIGURE 90. **SD16 A/D Channel 8 Input Amplifier**

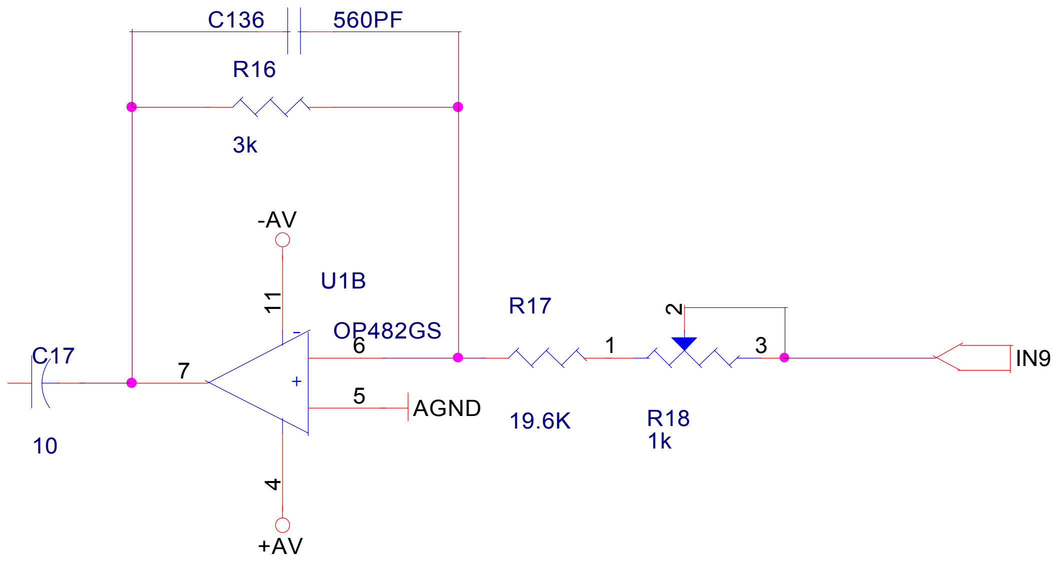


FIGURE 91. **SD16 A/D Channel 9 Input Amplifier**

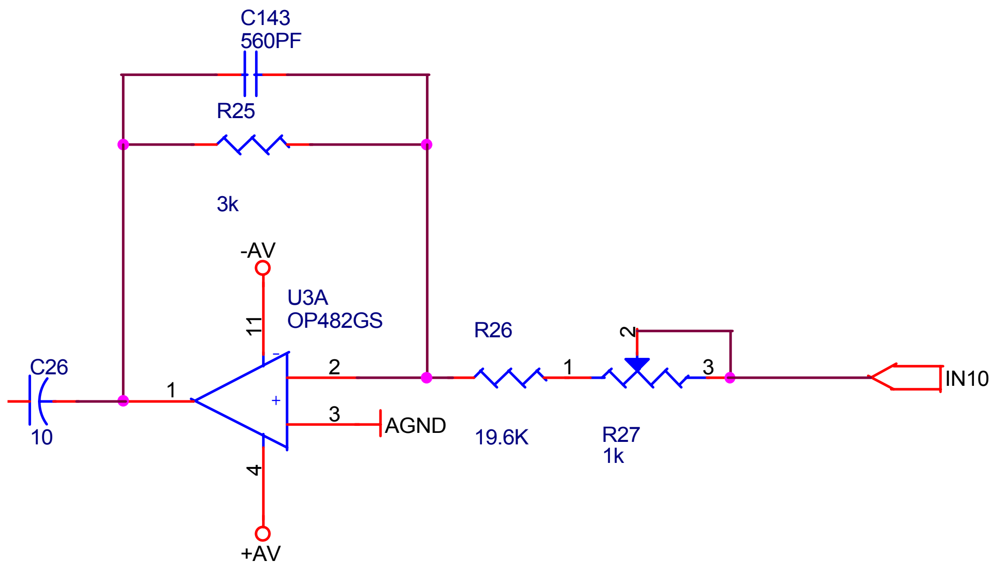


FIGURE 92. **SD16 A/D Channel 10 Input Amplifier**

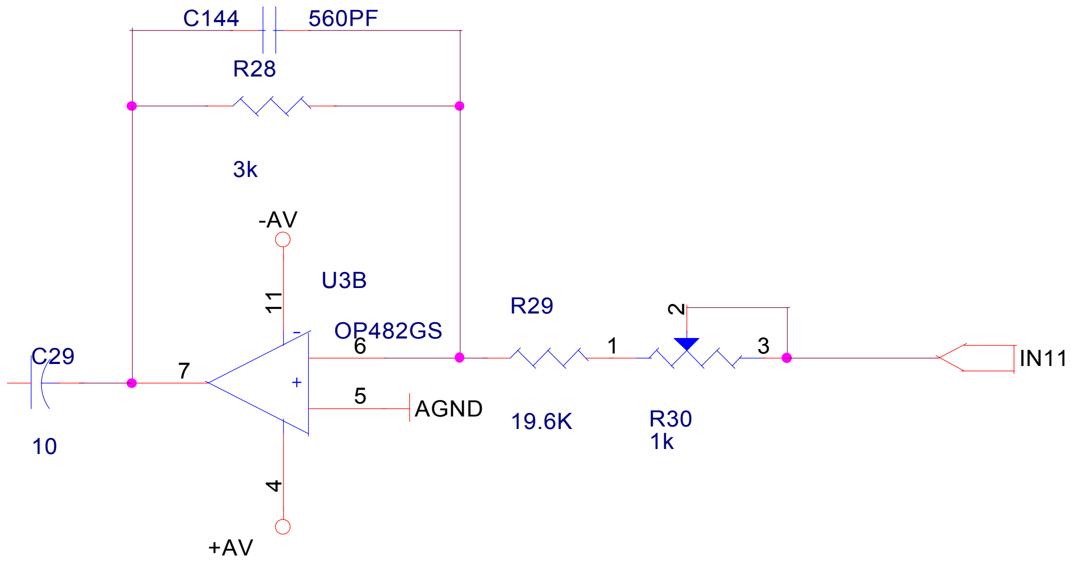


FIGURE 93. **SD16 A/D Channel 11 Input Amplifier**

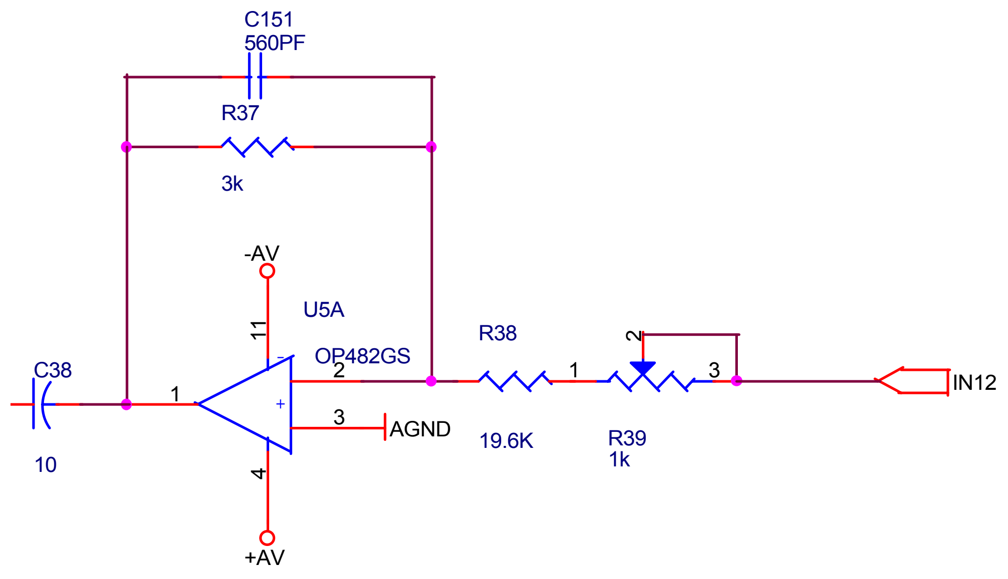


FIGURE 94. **SD16 A/D Channel 12 Input Amplifier**

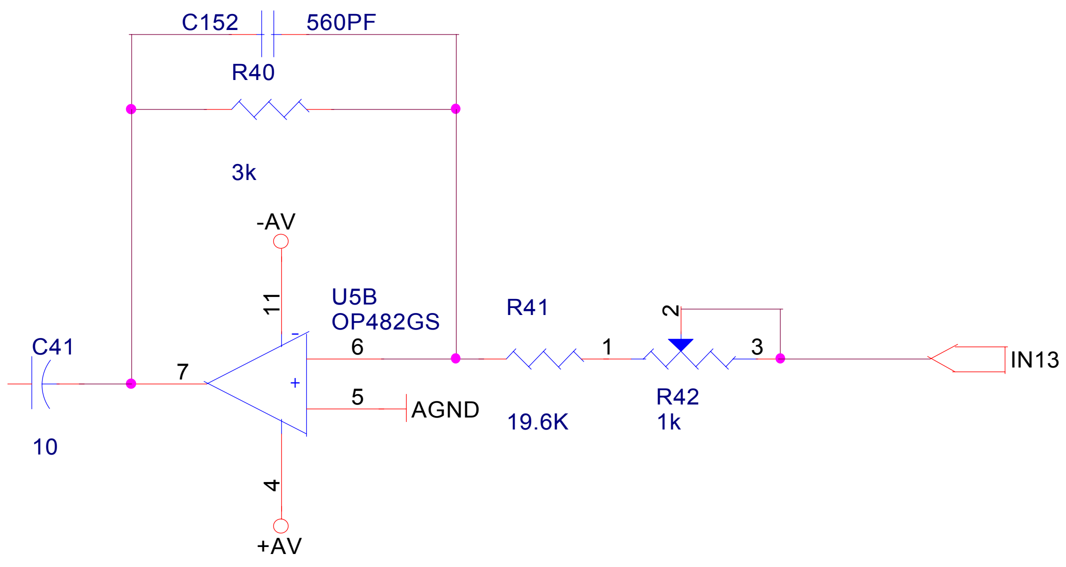


FIGURE 95. **SD16 A/D Channel 13 Input Amplifier**

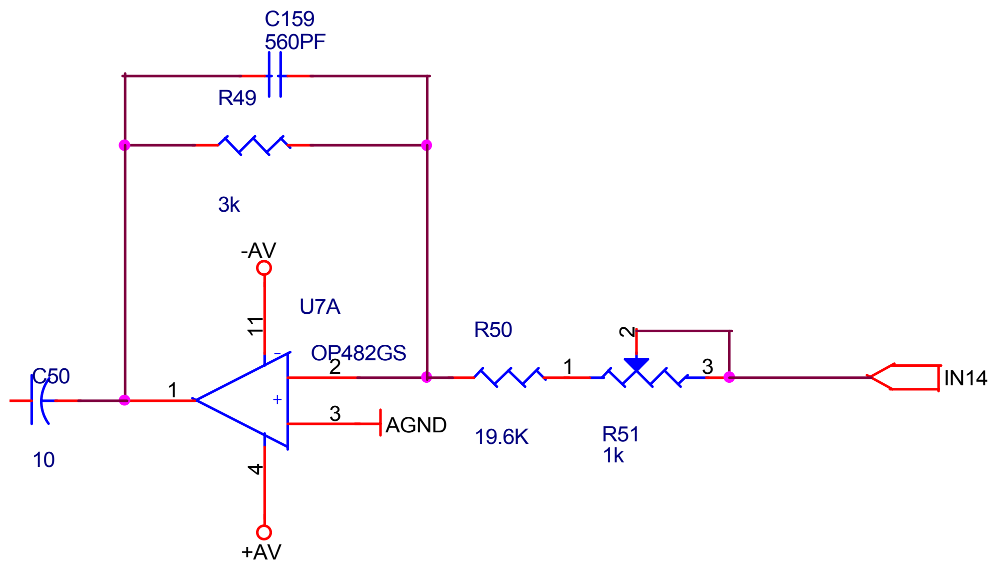


FIGURE 96. SD16 A/D Channel 14 Input Amplifier

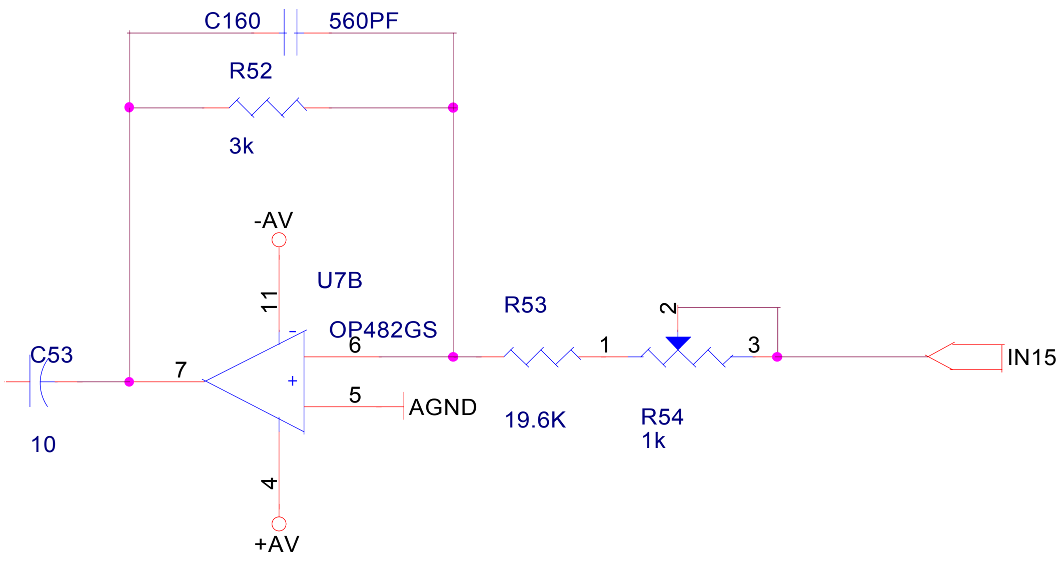


FIGURE 97. SD16 A/D Channel 15 Input Amplifier

The amplifier output signals are AC coupled to the converter inputs to remove any DC information or offset error from the audio signal. Each converter's serial output data is shifted into an independent onboard serial-to-parallel converter and held in a buffer register for delivery to the OMNIBUS host.

Sample data returned by the A/D converters are in 18 bit two's complement format. The non-inverting analog inputs result in return codes in the range of [131071, -131072]. Data on the upper 14 bits of the 32-bit read must be masked, as these bits are not driven by hardware on a read by the OMNIBUS host of the A/D holding registers. Logically shifting the 32-bit read value up 14 bits and then arithmetically shifting down 14 bits converts the masked 18-bit number to a sign-extended 32-bit number.

Audio Input Timing and Sample Rates

The A/D converters on the SD16 are controlled through a master clock, which, through divide down circuitry, determines the final sample rate of all sixteen of the input channels. To establish a particular sample frequency on the SD16, it is necessary to calculate an appropriate master clock frequency and program the host board DDS timebase to generate the required frequency.

The master clock rate is always 384 times the required sample rate. Master clock frequencies for several standard audio sample rates are given in the table below: other frequencies may be calculated simply by taking the required audio sample rate and multiplying by 384. Any sampling frequency from 3 kHz to 48 kHz may be used. Sampling at lower than 3 kHz will result in loss of accuracy and is not recommended.

Audio Sample Rate (kHz)	Required Master Clock (DDS) Frequency (MHz)
3	1.1520
32	12.2880
44.1	16.9344
48	18.4320

TABLE 110. SD16 Example Audio Input Sample Rates with Master Clock Frequencies

Sampling begins immediately following the DDS programming sequence. Both input and output are free-running once the timebase has been established. The A/D devices will convert continuously according to the sample rate determined by the master clock. It will continue to deliver data via the serial to parallel conversion hardware into the parallel holding registers regardless of whether or not the host CPU is retrieving data from the registers. There is no hardware provision for buffering or avoiding overflow conditions. Therefore, it is necessary to make sure that I/O bus host CPU software is responsive enough to read back all converted A/D data before the next conversion is complete.

Please note that it is not necessary to trigger conversion sequences in software, nor is there any supported means for triggering samples from external equipment.

Audio Input Range Trim

The SD16's input amplifiers have been trimmed at the factory and cannot be altered by the user.

Converter Reset/System Clock Control Register

The converter reset/system clock control register allows software to manipulate the reset control signal to the codecs and control the source of the signal used to generate the master clock for conversion timing and data transmission.

Bit definitions for the register are given below. Bit 0 is used to control the reset signal to all eight converters (only a single reset signal is generated from the logic on each SD16 module, which is connected to all eight codecs). Setting the bit low asserts reset active to the codecs, while setting the bit high deasserts reset. When reset is asserted, the D/A outputs go to mid value (0 volt output at the OMNIBUS I/O connector with the default output amplifier configuration) and the data values are undefined. This bit defaults to zero to place the codecs in reset following host board powerup or reset.

Bit 1 of the register controls the source of the master clock used to time the conversion system. Setting the bit low selects the OMNIBUS clock as the master clock (exact frequency depends on the host board and processor speed), while setting the bit high selects the DDS output clock from the OMNIBUS connector. This bit defaults to zero to select the OMNIBUS clock following host board powerup or reset.

Audio Output

The SD16 also implements sixteen channels of 18-bit digital to analog conversion via eight stereo D/A converters. Similar to the A/D converters, these serial converters are synchronously triggered by a single master clock that is provided by the OMNIBUS host board's direct digital synthesizer. This master clock is then divided down internally by internal logic and used to generate a serial bit clock and a left/right channel clock for the serial interface. A serial/parallel converter in onboard logic serves as an interface between the parallel I/O bus and the serial interface to the D/A devices.

Line level audio signals are generated by the SD16 from the D/A device's outputs. DC offsets are subtracted and the resultant AC signals filtered and summed to generate the line level outputs. The output circuit schematic is shown below (the circuit is repeated sixteen times, once for each output).

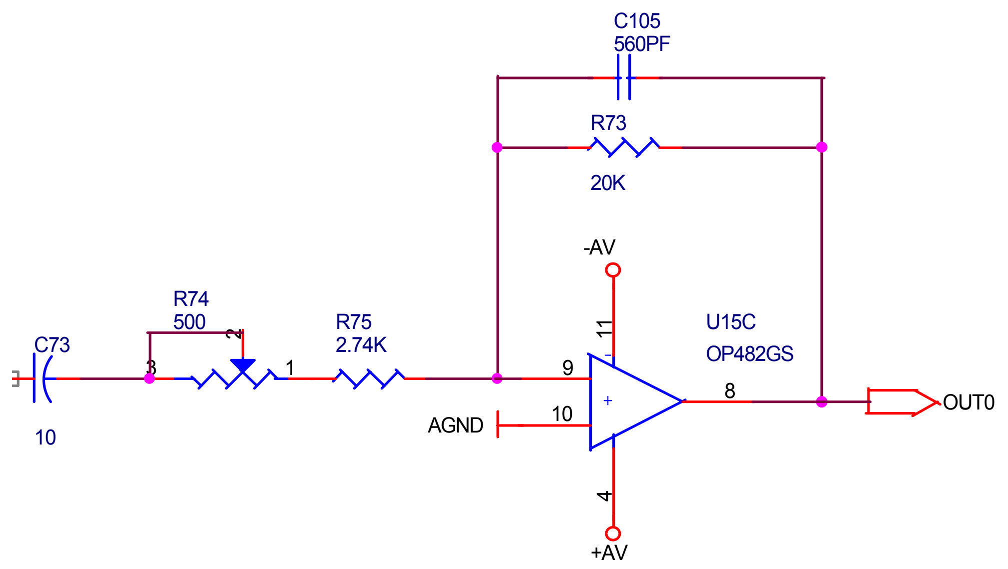


FIGURE 98. SD16 D/A Channel 0 Output Amplifier

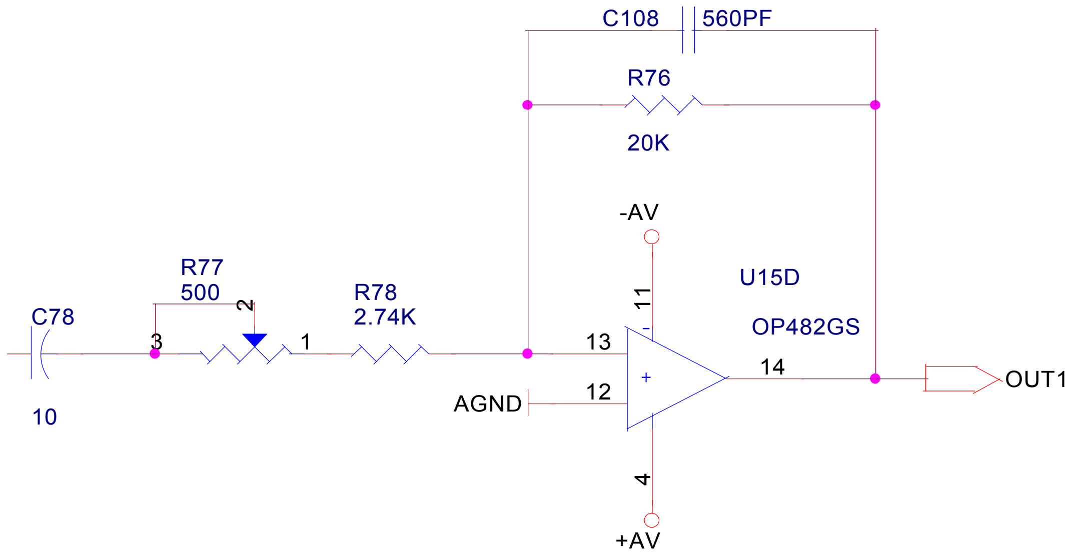


FIGURE 99. SD16 D/A Channel 1 Output Amplifier

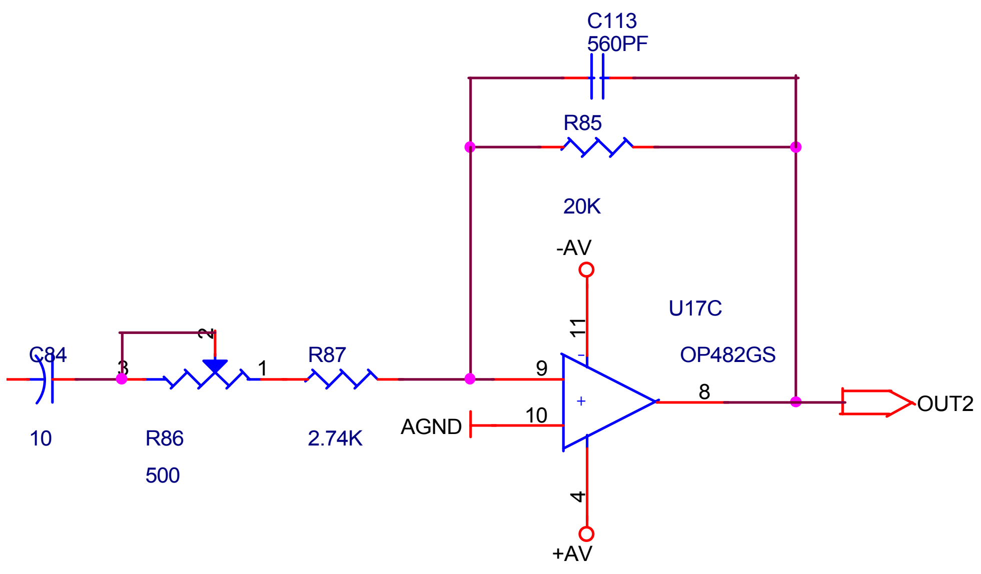


FIGURE 100. SD16 D/A Channel 2 Output Amplifier

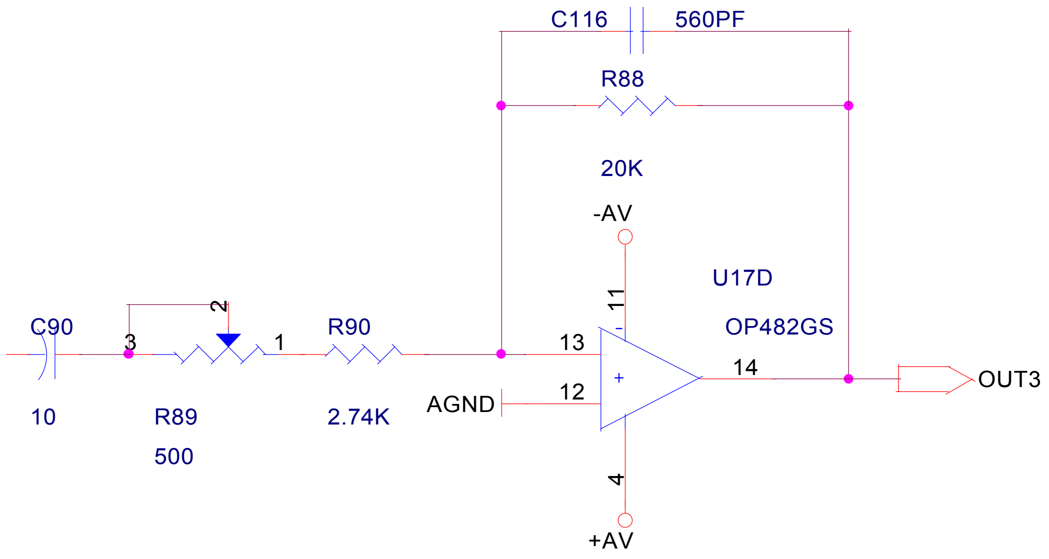


FIGURE 101. SD16 D/A Channel 3 Output Amplifier

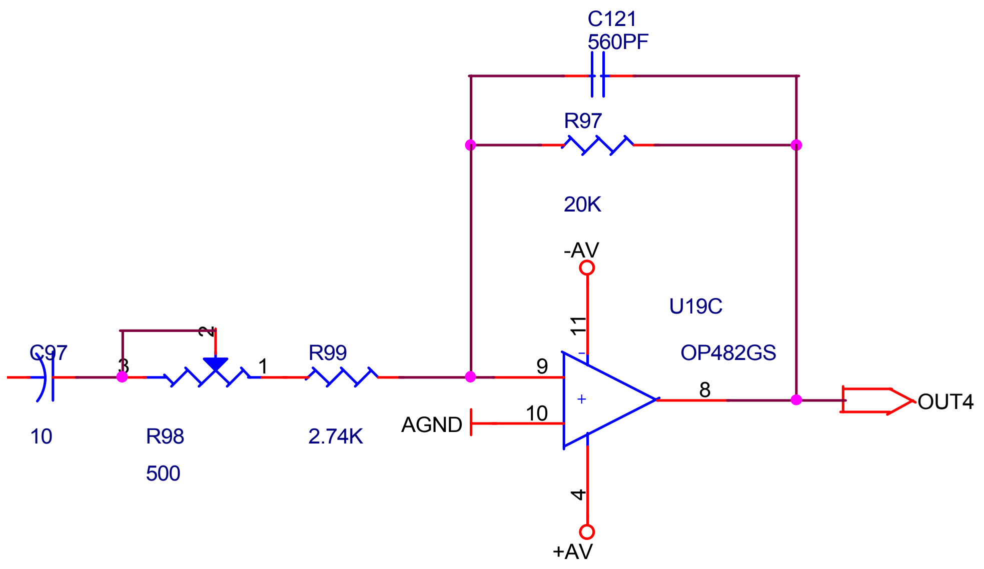


FIGURE 102. **SD16 D/A Channel 4 Output Amplifier**

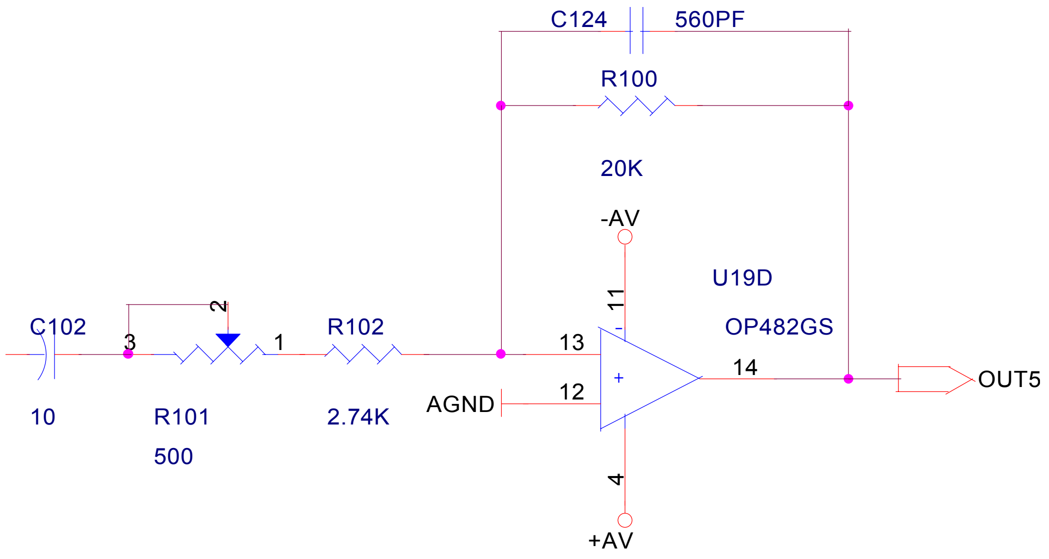


FIGURE 103. **SD16 D/A Channel 5 Output Amplifier**

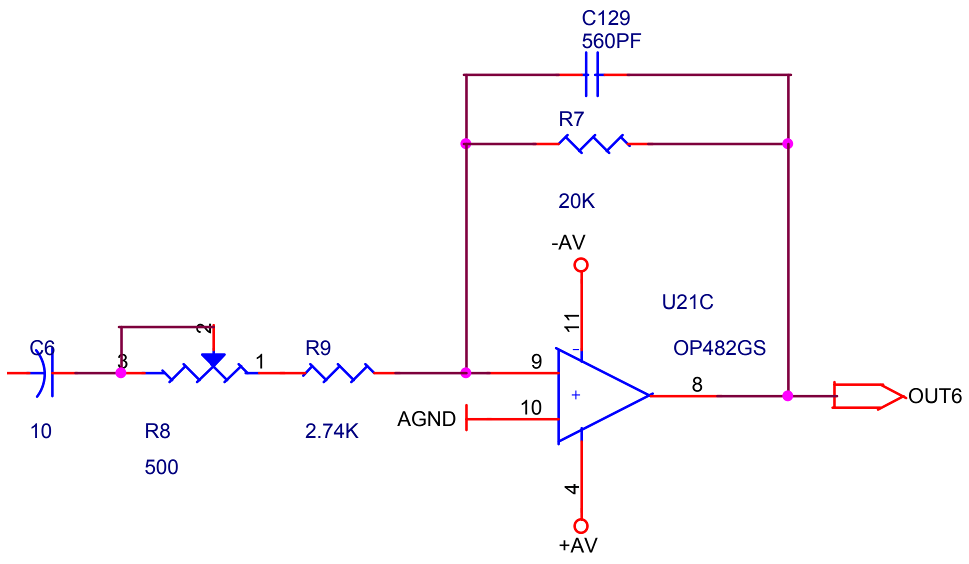


FIGURE 104. SD16 D/A Channel 6 Output Amplifier

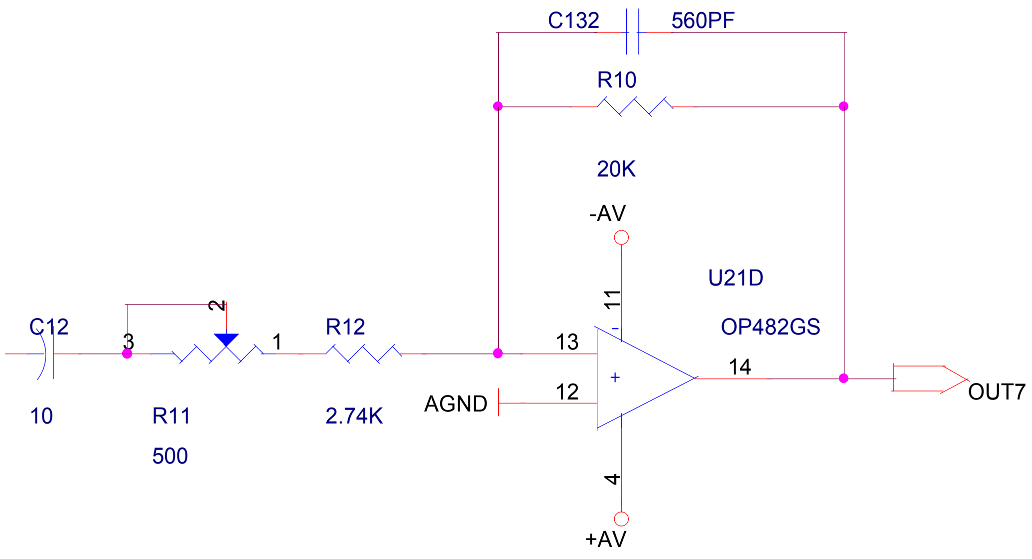


FIGURE 105. SD16 D/A Channel 7 Output Amplifier

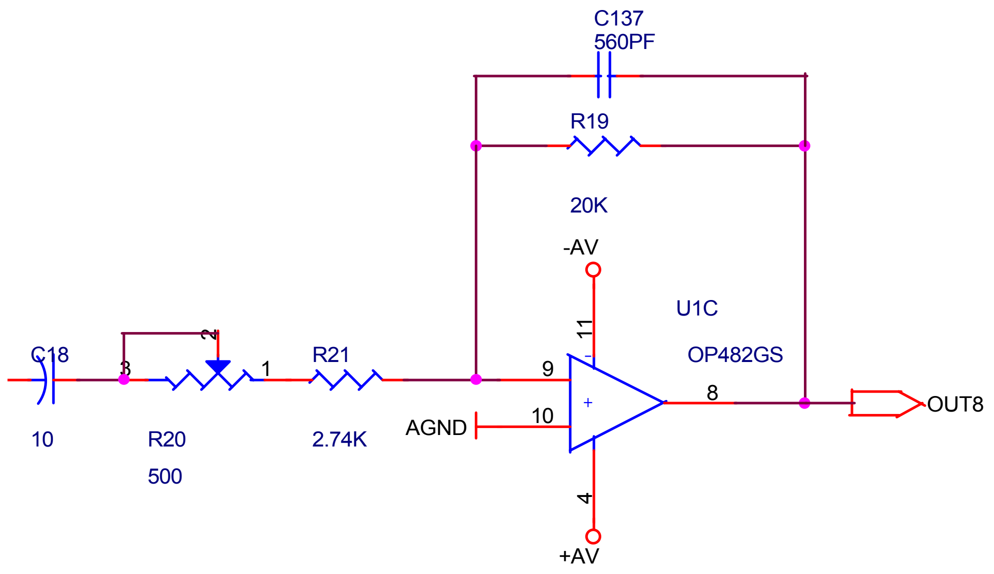


FIGURE 106. SD16 D/A Channel 8 Output Amplifier

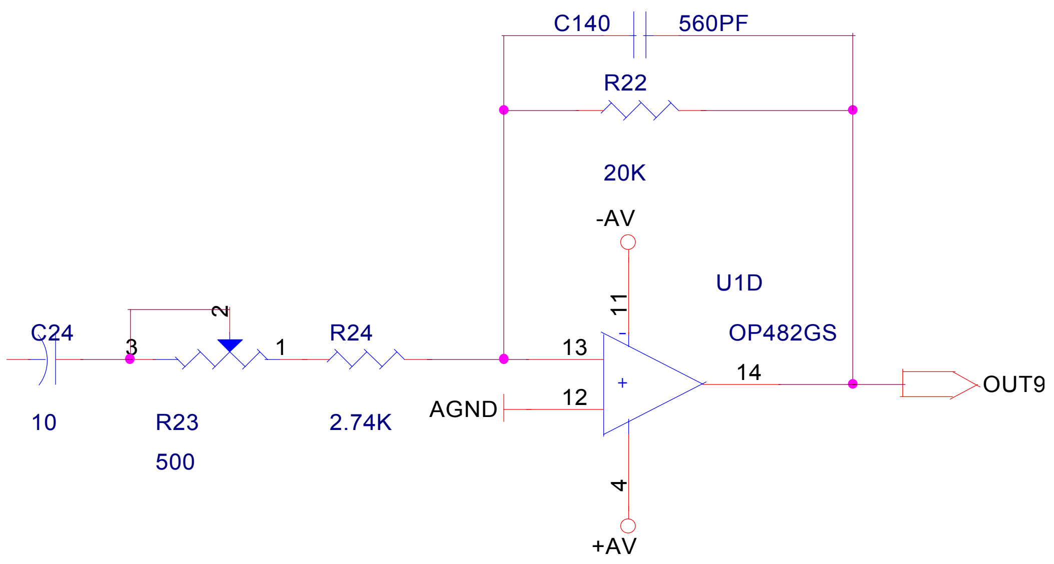


FIGURE 107. SD16 D/A Channel 9 Output Amplifier

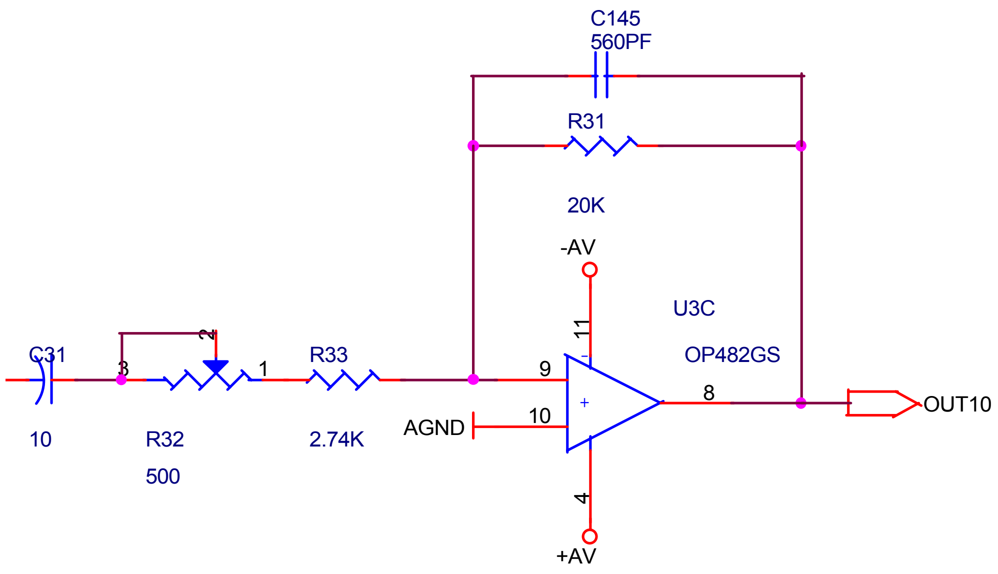


FIGURE 108. **SD16 D/A Channel 10 Output Amplifier**

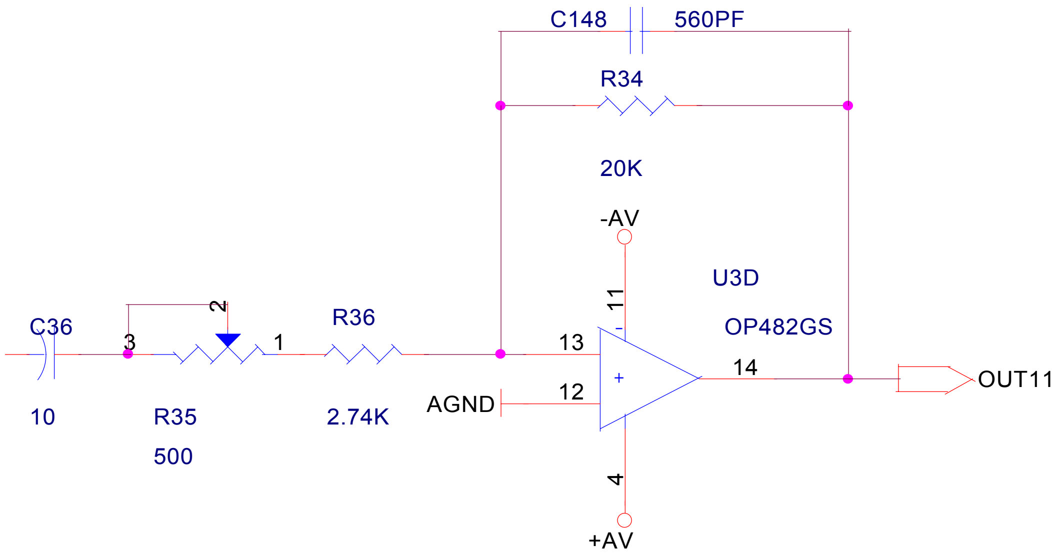


FIGURE 109. **SD16 D/A Channel 11 Output Amplifier**

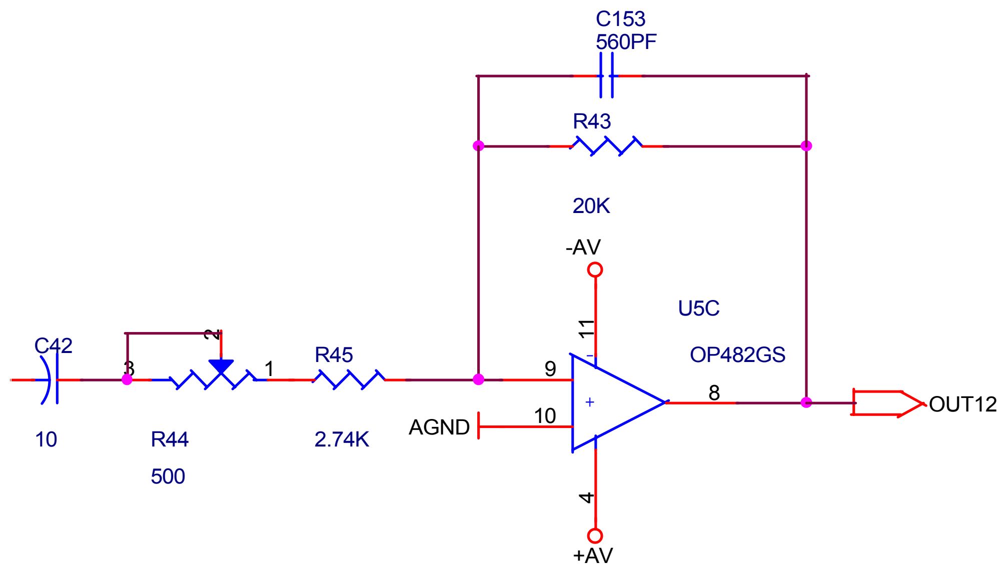


FIGURE 110. **SD16 D/A Channel 12 Output Amplifier**

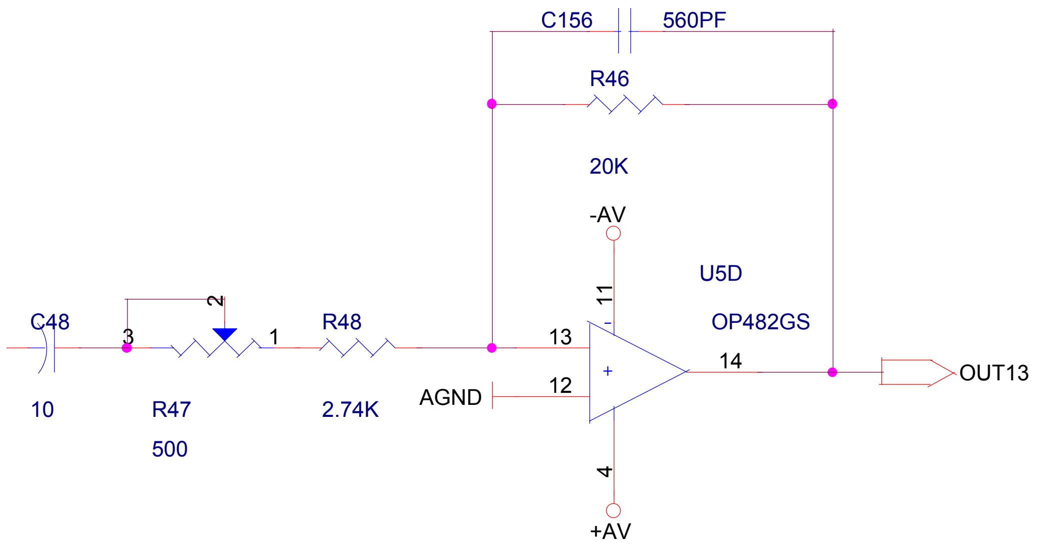


FIGURE 111. **SD16 D/A Channel 13 Output Amplifier**

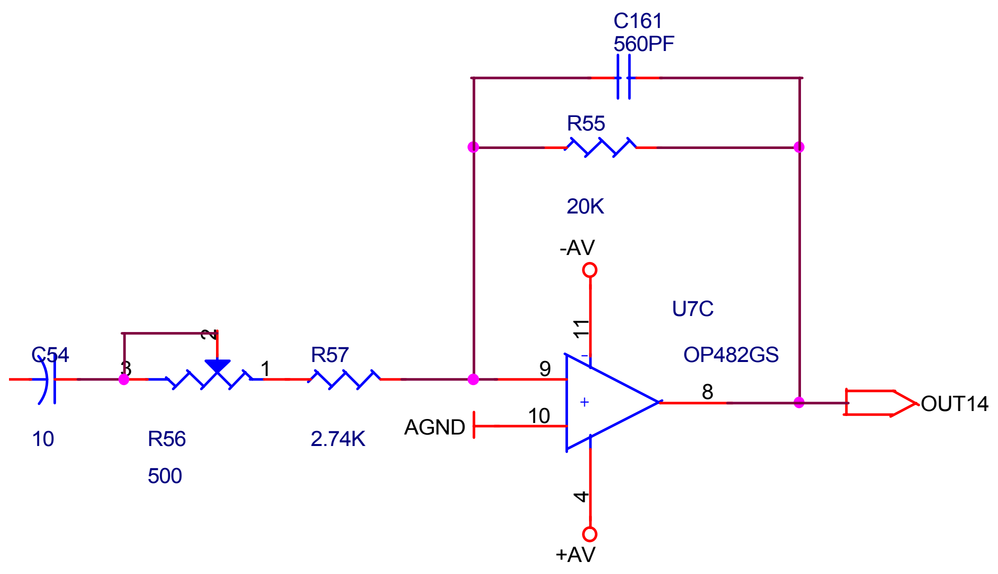


FIGURE 112. SD16 D/A Channel 14 Output Amplifier

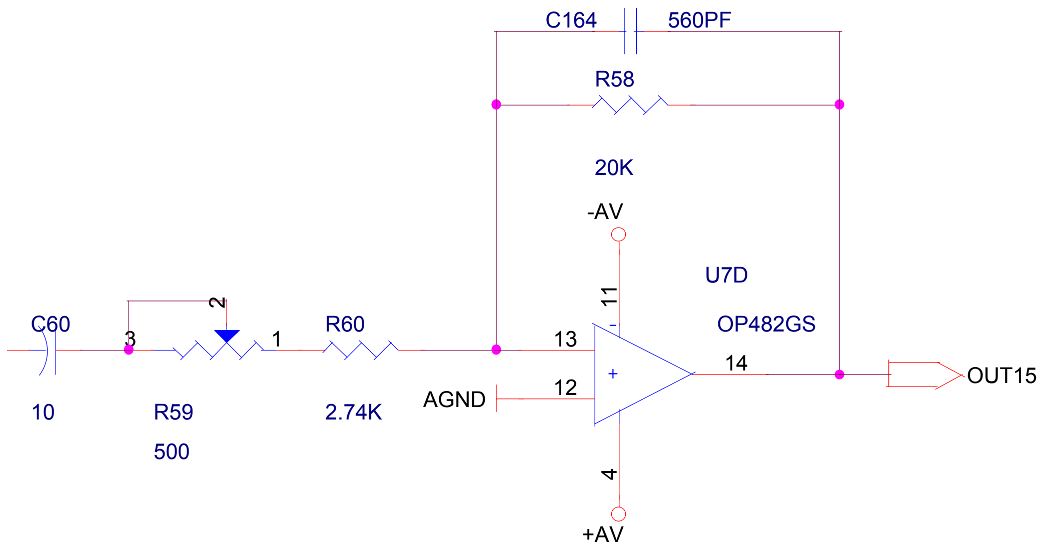


FIGURE 113. SD16 D/A Channel 15 Output Amplifier

The D/A converter interface expects input digital data in the form of two's complement numbers in the range of [131071, -131072]. The interface is compatible with direct host processor writes of 32-bit signed integers.

Audio Output Timing and Sample Rates

Industry standard audio sampling rates are also supported by the output circuitry of the SD16. Similar to the input electronics, the outputs are driven by a master clock that is nominally 384 times the frequency of the required sample rate. The identical DDS timer frequencies used for the A/D timing are also used for D/A conversion.

Audio Output Range Trim

The SD16's output amplifiers have been trimmed a the factory and cannot be adjusted by the user.

Initialization Issues

Please note the following SD16 initialization order. These rules apply after either a system powerup or a hardware reset is applied to the OMNIBUS host board, or whenever software needs to change the DDS synthesizer's output frequency in order to change the codec sampling frequency.

1. Set the codec reset /clock control register to 0 to place the codecs in reset and switch to the OMNIBUS clock as the codec master clock source.
2. Set the DDS to the desired master clock rate.
3. Write 0x2 to the codec control register to switch to the DDS as a master clock source.
4. Wait a minimum of seven seconds to allow for codec calibration and output settling.
5. Write 0x3 to the codec control register to deassert reset to the codecs with the DDS selected as the master clock.

CHAPTER 15

Servo16 Module

Module Introduction

The Servo16 module is designed for analog measurement and control applications requiring high accuracy and channel density. This module employs sixteen simultaneous sampling 100 kHz 16 bit A/Ds and sixteen simultaneous update 100 kHz 16 bit D/As for an unprecedented analog I/O density. Applications include vibration measurement and control, SONAR, and industrial process control. The simultaneous sampling capability is ideal for state-space or MIMO control systems.

Each analog input utilizes a Burr Brown ADS8321 A/D converter, a 4-pole analog anti-alias filter, a differential +/-10 Volt input for noise rejection and is DC accurate. Software programmable gain and offset error correction capability is implemented in onboard logic, with nonvolatile storage of calibration coefficients.

The sixteen output channels utilize the Analog Devices AD5544 D/A converter and also offer software programmable gain and offset error correction capability in onboard logic, with nonvolatile storage of coefficients. The output channels are cleared to 0 Volts at power up or upon reset and have a ±10 Volt output range. The module is also equipped with the ability to simultaneously update all the channels with a single command.

The Servo16 is ideal for applications where high channel count A/D and D/A are required such as simultaneous signal acquisition or playback, servo controls, state space control systems, and multichannel data recorders.

The following block diagram shows the features of the Servo16.

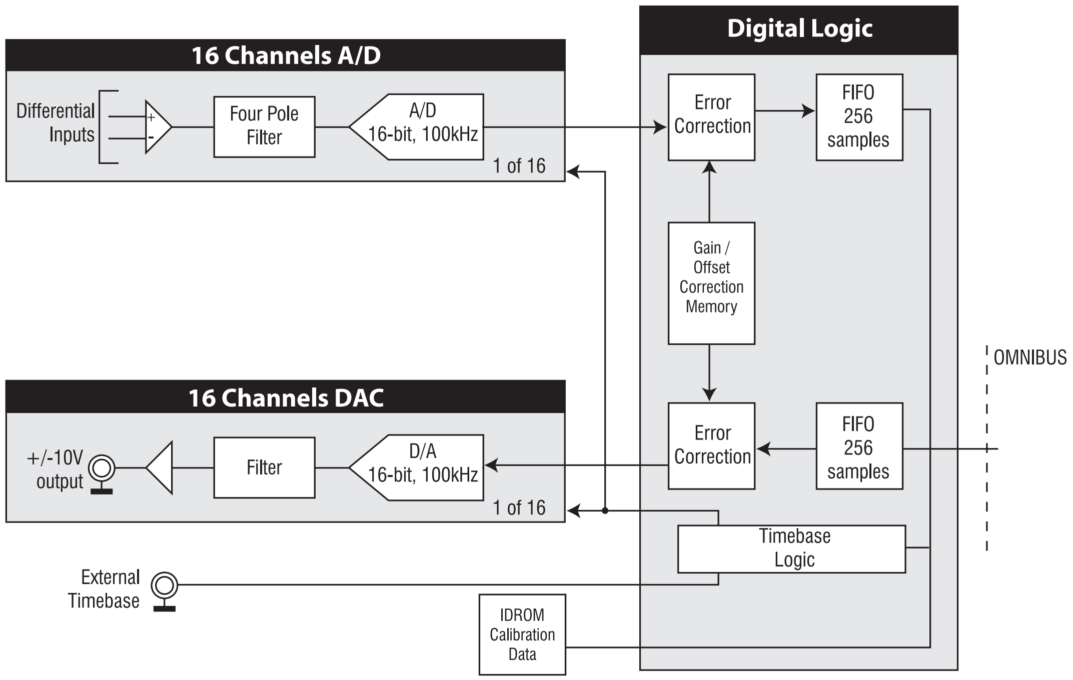


FIGURE 114. Servo16 Block Diagram

Bus Type:	Compatible with all OMNIBUS host products; Consumes one interrupt to host baseboard.	**Filter Characteristics:**	4-pole elliptic filter 3dB set at 50 kHz.
Power Requirements:	+5V analog 30 mA; +15V 120 mA; -15V 120 mA; +5V 250 mA	**Conversion Trigger Sources:**	host timers.
Physicals:	OMNIBUS mezzanine card; 2.0” X 4.6”	**Interface to Host:**	Memory-mapped 32 bit result returned for each pair. Input FIFOs hold up to 512 samples.
A/D Converters	16 Burr Brown ADS8321 successive approximation converters.	**D/A Converters**	16 Analog Devices AD5544 converters.
Resolution:	16-bit	**Resolution:**	16-bit
Update Rate:	1-100 kHz. (Maximum)	**Output Range:**	+/- 10V (custom ranges may be special ordered)
Analog Input Range:	+/- 10V differential (custom inputs may be special ordered)	**Settling Time:**	2 usec
S/N:	85 dB	**THD:**	-73 dB
THD:	-80dB	**Gain and Offset Error:**	Factory calibrated error correction coefficients programmable into FPGA.
Gain and Offset Error:	Factory calibrated error correction coefficients programmable into FPGA.	**Differential Nonlinearity Error:**	+/- 1.5 LSB
Differential Linearity Error:	+2.5/-0.5 LSB	**DAC Glitch Energy:**	45 nV-sec typ at MSB transition.
Input Impedance:	10 M ohm | | 5 pF	**Trigger Resources:**	DSP, timers or externally triggered.
Input Type:	Differential	**Interface to Host:**	Memory-mapped 32 bit number output for each pair. Output FIFOs hold up to 512 samples.

The target Peripheral Library provides support for each of the Servo16 functions, and the following sections give descriptions of the support routines. Refer to the **Servo16adc.h** and **Servo16gain.h** files located in the `c:/<target board>/Include/Target/Analog` director and **Omnibus.hlp or Zuma.hlp** Windows helpfile for complete details on each function.

Function Name	**Operation**
SERVO16_bleed_adc_fifo();	Read multiple A/D sample pairs from FIFO.
SERVO16_enable_adc_pair();	Specify active A/D channel pairs.
SERVO16_enable_dac_pair();	Specify active D/A channel pairs.
SERVO16_fill_dac_fifo();	Copy buffer contents to D/A FIFO.
Servo16_convert_to_gain()	Convert float to internal fixed-point format.
SERVO16_from_fixed();	Convert gain coef from internal format to IEEE.
SERVO16_get_adc_gain();	Read gain coef for specified A/D channel.
SERVO16_get_adc_offset();	Read offset coef for specified A/D channel.
SERVO16_get_dac_gain();	Read gain coef for specified D/A channel.
SERVO16_get_dac_offset();	Read gain coef for specified D/A channel.
SERVO16_read_adc_pair();	Read sample pair from A/D FIFO.
SERVO16_read_idrom();	Read entire module IDROM into buffer.
SERVO16_reset();	Reset all module peripherals.
SERVO16_run();	Begin data acquisition, waveform playback.
SERVO16_set_adc_fifo_threshold();	Set interrupt threshold for A/D FIFO.
SERVO16_set_adc_gain();	Set gain coef for specified A/D channel.
SERVO16_set_adc_offset();	Set offset coef for specified A/D channel.
SERVO16_set_dac_fifo_threshold();	Set interrupt threshold for D/A FIFO.
SERVO16_set_dac_gain();	Set gain coef for specified D/A channel.
SERVO16_set_dac_offset();	Set gain coef for specified D/A channel.
SERVO16_stop();	Terminate acquisition/playback.
SERVO16_to_fixed();	Convert coef from IEEE format to internal format.
SERVO16_write_dac_pair();	Write sample pair to D/A FIFO.
SERVO16_write_idrom();	Write buffer to module IDROM.

TABLE 111. C Language Servo16 Functions

The Servo16 library functions can be divided into several groups:

1. A/D and D/A management.
2. Calibration control.
3. Interrupt selection.
4. Identification readback.

A/D Control

The Peripheral Library includes functions for transferring data from the A/D FIFO, triggering A/D conversions via a timer and setting up a hardware timer-based trigger source.

Prior to acquiring data from the Servo16, the module must be initialized. Initialization consists of the following steps:

1. Read the calibration coefficients from the module IDROM using **SERVO16_read_idrom**. Write these coefficient values into the channel-specific calibration registers using **SERVO16_set_adc_gain** and **SERVO16_set_adc_offset**. The analog output (D/As) must be initialized in a similar fashion.

2. Reset the module to its power-up state via a call to **Servo16_reset()**. Insure that the module is not capable of generating interrupts using **SERVO16_stop**.

3. Specify which A/D and D/A channels are to be active during data acquisition and playback using calls to **SERVO16_enable_adc_pair** and **SERVO16_enable_dac_pair**, respectively.

4. Specify the fullness level of the FIFO above which the module will signal an interrupt to the baseboard via a call to **Servo16_set_adc_fifo_threshold** and **Servo16_set_dac_fifo_threshold.**

5. Clear the baseboard interrupt associated with the Servo16 module via a call to the Zuma functions **ClearInterrupt** and **ActivateInterrupt**. Then, route the module interrupt to the DSP via **InterruptSource**. Install the interrupt vector via a call to **InstallIntVector**. Finally, enable the DSP interrupt via **EnableInterrupt**.

The Servo16 module utilizes a single receive FIFO into which all samples from all channels are stored during the acquisition process. Eight pairs of A/D converters are available on the module. Each pair presents 32-bit data to the FIFO. The low-order sixteen bits of the 32-bit data corresponds to the converted results from the lower-numbered A/D in the pair. The high-order 16-bits of the 32-bit data corresponds to the converted results from the higher-numbered A/D in the pair. Only the sample results from A/D pairs, which are currently enabled, are stored in the FIFO during acquisition. Individual channel pairs are enabled via the **Servo16_enable_adc()** function.

During acquisition, sample pairs are added into the FIFO starting with the lowest-numbered, enabled A/D pair and ending with the highest-numbered, enabled A/D pair. Thus, the data stored in the FIFO consists of data received from *active* A/D channel pairs only. For each conversion trigger received by the module, one 32-bit value from each enabled channel pair is stored into the FIFO. The term *sweep* is used within this documentation to denote the collection of 32-bit sample pairs stored into the FIFO for each trigger event.

By like manner, the Servo16 module utilizes a single transmit FIFO from which all samples to all D/A channels are stored during the playback process. Eight pairs of D/A converters are available on the module. Each pair extracts 32-bit data from the FIFO. The low-order sixteen bits of the 32-bit data corresponds to the converted results from the lower-numbered D/A in the pair. The high-order 16-bits of the 32-bit data corresponds to the converted results from the higher-numbered D/A in the pair. Only the samples from active D/A pairs are extracted from the FIFO during playback. Individual channel pairs are enabled via the **Servo16_enable_dac()** function.

During playback, sample pairs are extracted from the FIFO starting with the lowest-numbered, enabled D/A pair and ending with the highest-numbered, enabled D/A pair. Thus, the data extracted from the FIFO consists of data sent to *active* D/A channel pairs only. For each conversion trigger received by the module, one 32-bit value from each enabled channel pair is extracted from the FIFO. The term *sweep* is used within this documentation to denote the extraction of 32-bit sample pairs from the FIFO for each trigger event.

Acquisition and/or playback is enabled via the **Servo16_start()** function. Afterwards, 32-bit sample data for all enabled A/D pairs are added to the FIFO each time the A/Ds are triggered via the conversion clock. All enabled A/Ds are always simultaneously triggered each time a trigger pulse in received by the module. Simultaneously, 32-bit sample data for all enabled D/A pairs are extracted from the FIFO each time the D/As are triggered via the conversion clock. All enabled D/As are always simultaneously triggered each time a trigger pulse in received by the module

When the FIFO reaches the previously programmed fullness level, the module can assert an interrupt to the baseboard. The baseboard can be programmed to bleed data from the A/D FIFO and to insert data into the D/A FIFO either using an interrupt handler or via DMA.

Within an interrupt handler, use the **Servo16_read_adc_pair()** function to read the oldest channel pair result from the FIFO. Repeat this call for as many pairs as have been enabled in order to remain aligned on sweep boundaries.

Alternately, application software may call the **Servo16_bleed_adc_fifo()** function to bleed multiple channel pairs into a memory buffer using a specified DMA channel. This usually results in superior real-time performance.

By like manner, use the **Servo16_write_dac_pair()** function to store fresh sample pairs into the output FIFO. Repeat this call for as many pairs as have been enabled in order to remain aligned on sweep boundaries.

Alternately, application software may call the **Servo16_fill_dac_fifo()** function to store data for multiple channel pairs from a memory buffer into the FIFO using a specified DMA channel. This usually results in superior real-time performance.

Calibration

The Servo16 has been architected without using trimpots for calibration of A/D and D/A scale factors and offsets. Instead, the onboard logic mathematically corrects the results of each analog input channel according to the formula $y = mx + b$ where m is the scalefactor (gain) to apply to the input channel and b is the offset to apply to the input channel. Separate gain and offset values are supported for each analog input channel.

During factory calibration, the optimal coefficients for the gain and offset for each channel have been measured and stored into the flash ROM onboard the Servo16 module. These coefficients must be retrieved at runtime via the **Servo16_read_idrom()** function and stored into the Servo16 in order to obtain measurements within the factory-specified accuracy of the module.

The **Servo16_set_adc_gain()** and **Servo16_set_adc_offset()** functions are available for use in application software to programmatically adjust the gain and offset of all channels of the module. Again, symmetric **Servo16_set_dac_gain()** and **Servo16_set_dac_offset()** functions are available for the D/A outputs.

Identification Readback

The **Servo16_read_idrom()** function is used to read the identification ROM on the module to retrieve factory calibration coefficients, check its identity and revision level. The function fills out an `Servo16_ID` structure with the information stored in the ID ROM on the module. The `Servo16_ID` structure is defined in the **omnibus.h** file and contains the following information:

1. Module name (the null-terminated string “Servo16”)
2. Module revision level
3. Calibration coefficients for gain and offset corresponding to each A/D channel
4. Checksum

This data is preprogrammed at the factory and should not be altered by the user.

Example Programs for the Servo16 Module

The following sections describe the example programs available in the Developer’s Package for use with a target DSP card with an Servo16 analog interface module installed.

SNAP. `SNAP` utilizes the target DSP’s Servo16 module analog input circuitry to sample an external signal and calculate statistics on the resulting digital data. This program is an example of how to handle interrupt driven A/D sampling at high rates, and can serve as a test program for determining the statistical noise performance of a single A/D channel.

`SNAP` uses the target DSP trigger matrix electronics to set up an external timer to trigger conversions on a selected A/D converter channel. As conversions are triggered, the A/D’s conversion complete interrupt causes an external interrupt on the target processor. This causes the interrupt handler to run, which retrieves the newly converted data and stores it to a buffer in processor memory. Once the buffer is full, the interrupt is deactivated and the program code proceeds to calculate the statistics variables on the gathered data.

For Codewright Users: For Codewright Users, the linker command (`.CMD`) files and Codewright project (`.PJT`) files necessary to rebuild the `SNAP` program are included with the Developer's Package.

For Code Composer Studio Users: For Code Composer Studio User, the make file (`.MAK`) file necessary to rebuild the `SNAP` program are included with the Developer's Package.

`SNAP` may be used as a basis for a custom data acquisition program, handling one or more A/D channels and either buffering the required data or passing the data to a host program via the card's bus mastering capability (see the section below for more information about host applications).

WAVE. `WAVE` is very similar in organization to the `SNAP` program, discussed above, and serves as an example of using the D/A converter outputs to generate signals based on digital signal data buffered in the DSP's memory.

The program first generate a scaled user-selected sine, square, or triangle wave table in memory, in a format suitable for use by the D/A devices. The program sets up an external timer as a timebase for D/A conversions, and initializes the target DSP's trigger matrix to direct the timebase output to the D/A devices. This causes the D/A's to continuously convert the values present in their data latches. Simultaneously, the conversion trigger also causes an analog interrupt handler to run, which feeds the next available data point from the sine wave buffer to the D/A's. This has the effect of creating a continuous analog sine wave on the D/A outputs, at the sample rate and resolution defined in the program.

As with the previous examples, the linker command and Code Composer Studio make files necessary to rebuild the `WAVE` program are included with the Developer's Package. `WAVE` can serve as the basis of a waveform generator, or in concert with the A/D input capability demonstrated in `SNAP` and `TEST`, can be used to generate feedback signals in a high-speed target DSP-based servo control system.

Memory Mapping

The Servo16 module functions are mapped according to the following table. The addresses listed are references to the C language address `#define` operators given in the **periph.h** file included with the host board's Development Package Peripheral Library. Each function is listed with a range of addresses in which the hardware is addressed. Bus width within these address ranges varies, and depends on the device and function being addressed (see below for details).

Function	Read/ Write	OMNIBUS Slot 0 Address	OMNIBUS Slot 1 Address	OMNIBUS Slot 2 Address
Data FIFO	R/W	IOMOD0 + 0x0	IOMOD4 + 0x0	IOMOD8 + 0x0
A/D Channel Enable Register	R/W	IOMOD0 + 0x1	IOMOD4 + 0x1	IOMOD8 + 0x1
A/D Control Register	W	IOMOD0 + 0x2	IOMOD4 + 0x2	IOMOD8 + 0x2
A/D FIFO Threshold	R/W	IOMOD0 + 0x5	IOMOD4 + 0x5	IOMOD8 + 0x5
D/A Channel Enable Register	R/W	IOMOD0 + 0x11	IOMOD4 + 0x11	IOMOD8 + 0x11
D/A Control Register	W	IOMOD0 + 0x12	IOMOD4 + 0x12	IOMOD8 + 0x12
D/A FIFO Threshold	R/W	IOMOD0 + 0x15	IOMOD4 + 0x15	IOMOD8 + 0x15
Run Control Register	W	IOMOD0 + 0x16	IOMOD4 + 0x16	IOMOD8 + 0x16
A/D Gain Coefficient Memory	R/W	IOMOD1 + 0x0..0xF	IOMOD5 + 0x0..0xF	IOMOD9 + 0x0..0xF
A/D Offset Coefficient Memory	R/W	IOMOD1 + 0x10..0x1F	IOMOD5 + 0x10..0x1F	IOMOD9 + 0x10..0x1F
D/A Gain Coefficient Memory	R/W	IOMOD2 + 0x0..0xF	IOMOD6 + 0x0..0xF	IOMOD10 + 0x0..0xF
D/A Offset Coefficient Memory	R/W	IOMOD2 + 0x10..0x1F	IOMOD6 + 0x10..0x1F	IOMOD10 + 0x10..0x1F
IDROM data	R/W	IOMOD3 + 0x0	IOMOD7 + 0x0	IOMOD11 + 0x0
IDROM clock	W	IOMOD3 + 0x1	IOMOD7 + 0x1	IOMOD11 + 0x1
SROM Programming Register	R/W	IOMOD3 + 0x4	IOMOD7 + 0x4	IOMOD11 + 0x4
SROM Programming Enable Register	W	IOMOD3 + 0x5	IOMOD7 + 0x5	IOMOD11 + 0x5

TABLE 112. Servo16 Memory Map

Interrupt Usage

The Servo16 module uses two interrupts to the host processor: one for input and one for output. These interrupts allow the application software to receive an interrupt when the FIFO in either direction has a pending transaction. In many applications, this is used to trigger a DMA data transfer or signal the processor to send or get data. The interrupt is asserted by the module when the FIFO level is above the threshold level for A/D FIFO, or below the threshold level for D/A FIFO as set by the respective FIFO threshold registers. The interrupts are active low level sensitive.

Interrupt	Function
0	A/D FIFO is above the A/D FIFO threshold level.
1	D/A FIFO is below the D/A FIFO threshold level.

TABLE 113. Servo16 Interrupt Usage

Pin Connector I/O

The Servo16 signal I/O connector is the standard OMNIBUS 50 pin, 0.050 pitch D connector (Amp P/N 173280). This table presents several pin lists according to the type of host card you may be using, since the output connectors on the host cards have different pin number schemes. Refer to the host card's *Instruction Manual*, *"Hardware"* for additional details on how to connect to the Servo16's I/O pins.

Servo16 Omnibus M173280-3 Connector Pin Number	ChicoPlus & Hombre Baseboard 100 Pin MDR Connector		cM44 & cM6x Baseboard SCSI-2 50 Pin Connector	M44, M6x, & SBC6x Baseboard IDC50 Connector	M44 & M6x Baseboard DB15 Connector	Function
	Module Site 0	Module Site 1				
1	100	75	25	1		Output 10
2	50	25	24	2		Output 6
3	99	74	23	3		Output 11
4	49	24	22	4		Output 7
5	98	73	21	5		Input 8 -
6	48	23	20	6		Input 9 -
7	97	72	19	7		Input 10 -
8	47	22	18	8		Input 11 -
9	96	71	17	9		Input 12 -
10	46	21	16	10		Input 13 -
11	95	70	15	11		Input 14 -
12	45	20	14	12		Input 15 -
13	94	69	13	13		Input 0 -
14	44	19	12	14		Input 1 -
15	93	68	11	15		Input 2 -
16	43	18	10	16		Input 3 -
17	92	67	9	17		Input 4 -
18	42	17	8	18		Input 5 -
19	91	66	7	19		Input 6 -
20	41	16	6	20		Input 7 -
21	90	65	5	21		Output 5
22	40	15	4	22		Output 9
23	89	64	3	23		Output 4
24	39	14	2	24		Output 8
25	88	63	1	25		Reserved
26	38	13	50	26		Output 14
27	87	62	49	27		Output 2
28	37	12	48	28		Output 15
29	86	61	47	29		Output 3
30	36	11	46	30		Input 8 +
31	85	60	45	31		Input 9 +
32	35	10	44	32		Input 10 +
33	84	59	43	33		Input 11 +
34	34	9	42	34		Input 12 +
35	83	58	41	35		Input 13 +
36	33	8	40	36	1	Input 14 +
37	82	57	39	37	9	Input 15 +
38	32	7	38	38	2	Input 0 +
39	81	56	37	39	10	Input 1 +
40	31	6	36	40	3	Input 2 +
41	80	55	35	41	11	Input 3 +
42	30	5	34	42	4	Input 4 +
43	79	54	33	43	12	Input 5 +
44	29	4	32	44	5	Input 6 +
45	78	53	31	45	13	Input 7 +
46	28	3	30	46	6	Output 1
47	77	52	29	47	14	Output 13
48	27	2	28	48	7	Output 0
49	76	51	27	49	15	Output 12
50	26	1	26	50	8	Analog Ground

TABLE 114. Servo16 I/O Connector Pinout

Functions

A/D Input Circuitry

Each A/D channel of the Servo16 is an independent signal chain composed of an instrumentation amplifier, analog filter and A/D converter. There are no multiplexed analog signals. The channels sample simultaneously for near zero channel to channel phase error (within nanoseconds) which is ideal for state space controls. Each channel has a high impedance differential input instrumentation amplifier for common mode noise rejection. This input amplifier is followed by a fourth-order active filter for anti-aliasing that is set at the factory to a 50 kHz break frequency (-3 dB). The following diagrams show the input circuitry for each A/D channel.

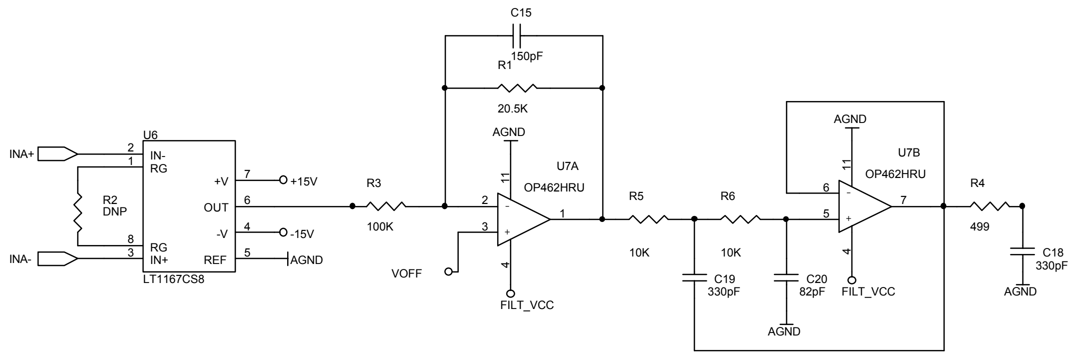


FIGURE 115. Servo16 Channel 0 Input Circuitry Schematic

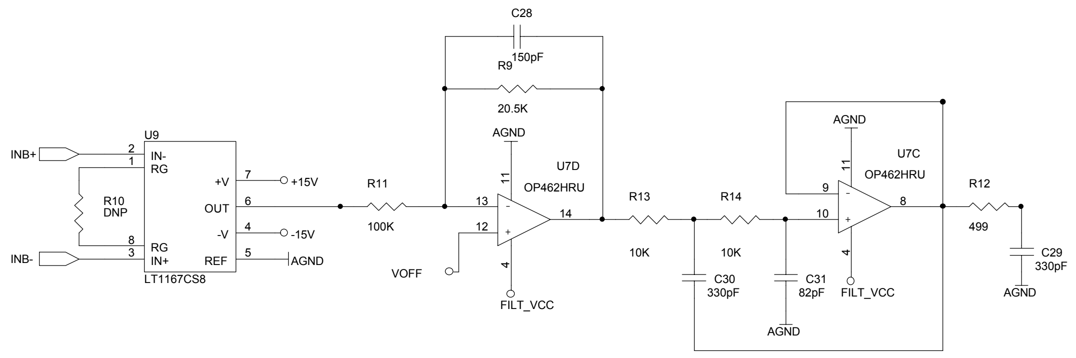


FIGURE 116. Servo16 Channel 1 Input Circuitry Schematic

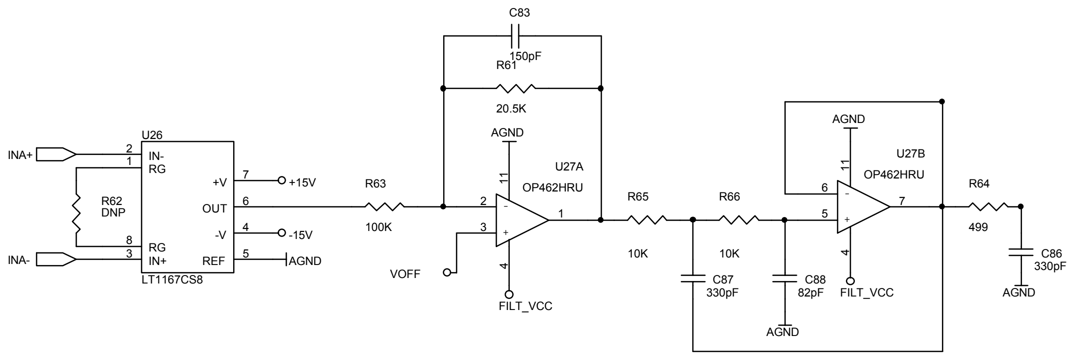


FIGURE 117. **Servo16 Channel 2 Input Circuitry Schematic**

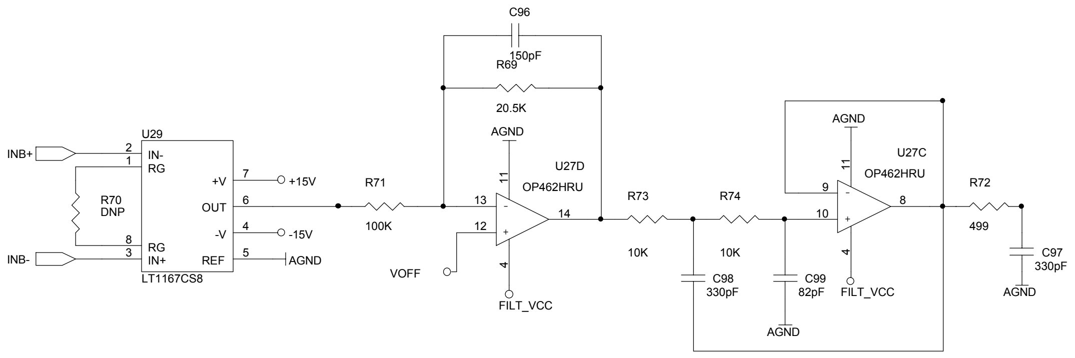


FIGURE 118. **Servo16 Channel 3 Input Circuitry Schematic**

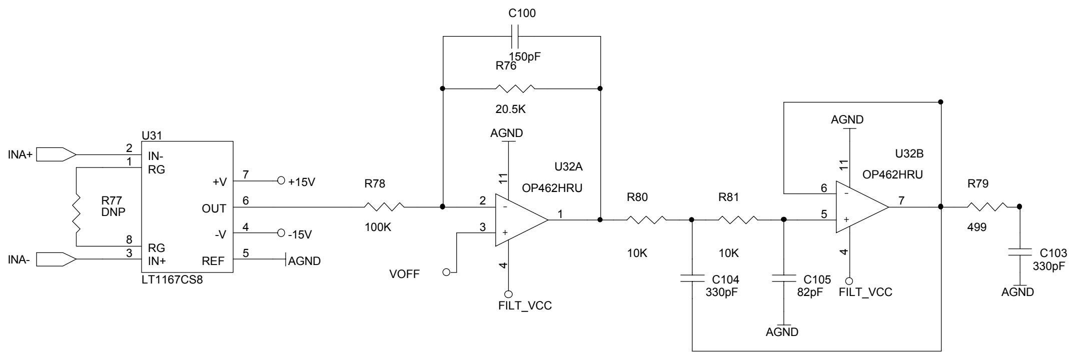


FIGURE 119. **Servo16 Channel 4 Input Circuitry Schematic**

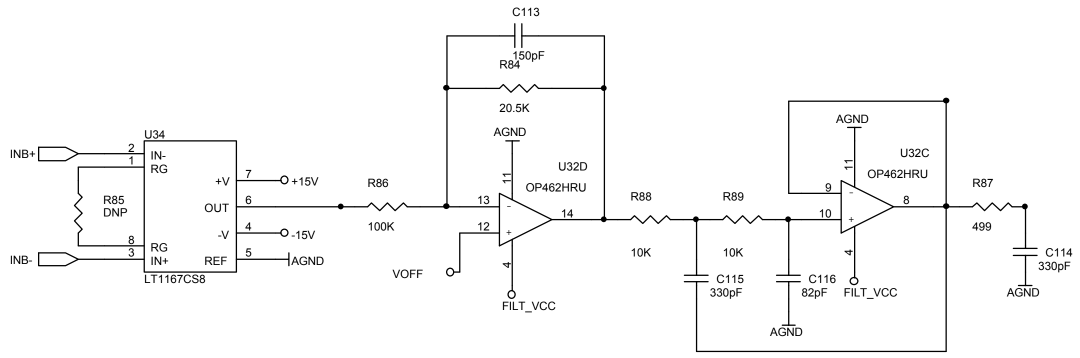


FIGURE 120. Servo16 Channel 5 Input Circuitry Schematic

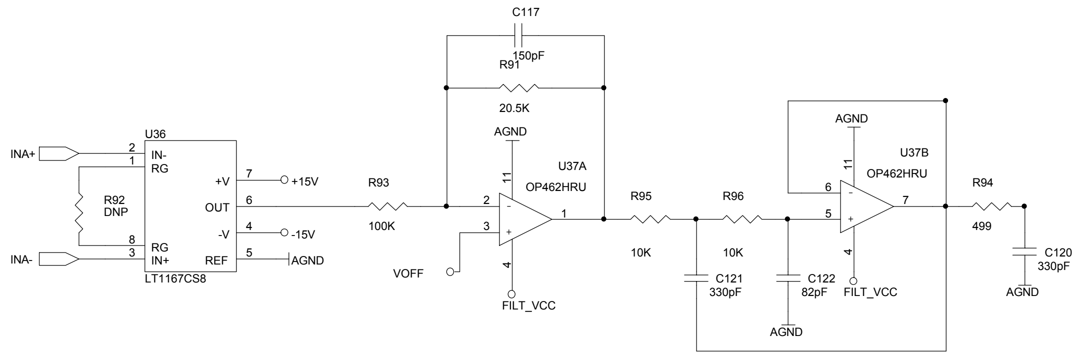


FIGURE 121. Servo16 Channel 6 Input Circuitry Schematic

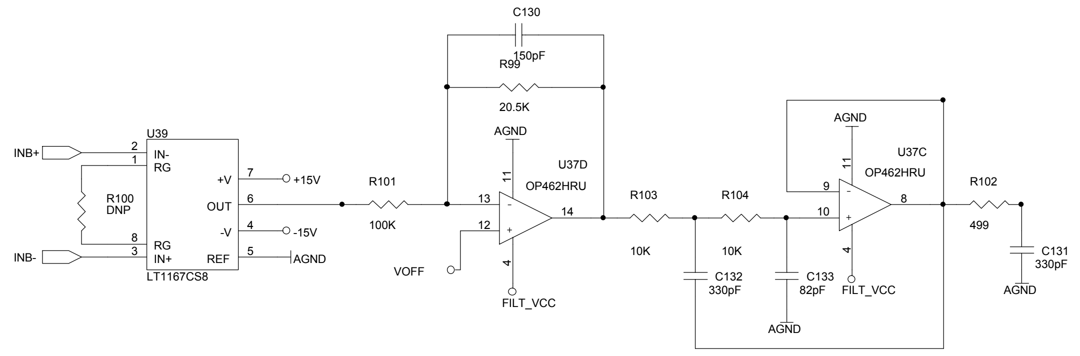


FIGURE 122. Servo16 Channel 7 Input Circuitry Schematic

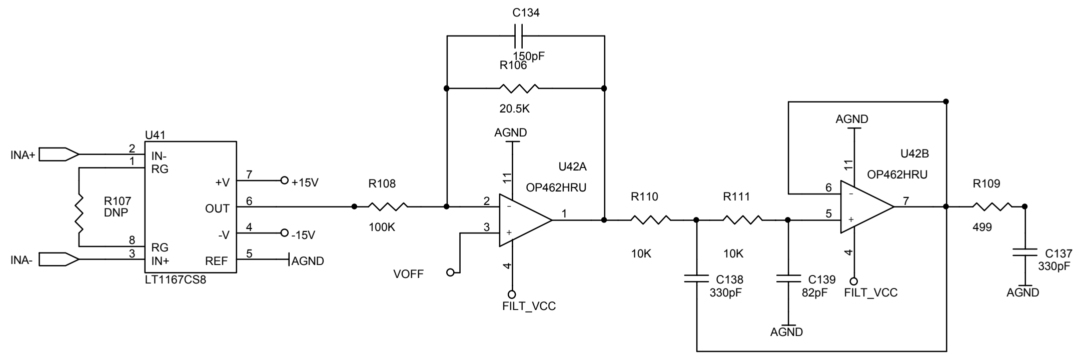


FIGURE 123. Servo16 Channel 8 Input Circuitry Schematic

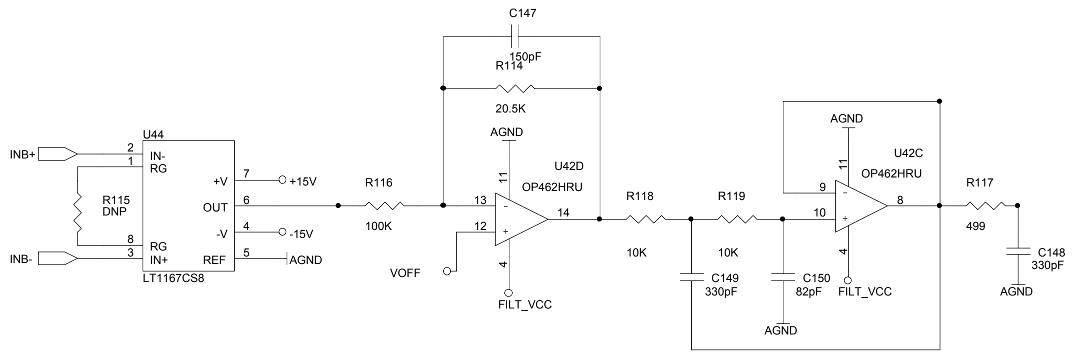


FIGURE 124. Servo16 Channel 9 Input Circuitry Schematic

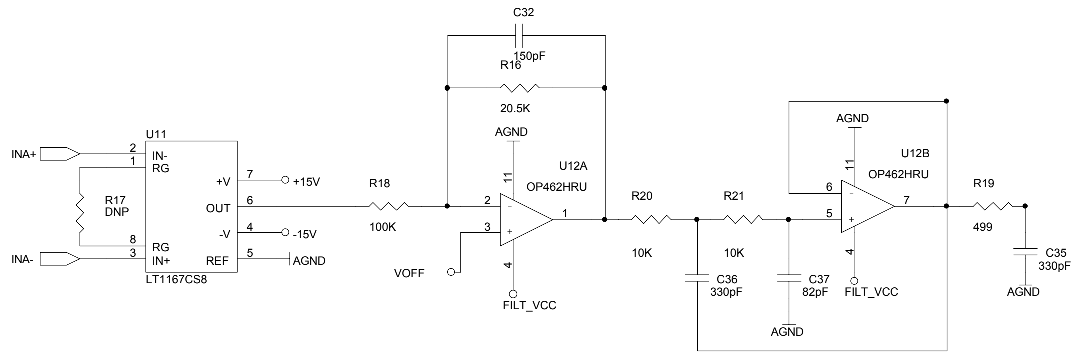


FIGURE 125. Servo16 Channel 10 Input Circuitry Schematic

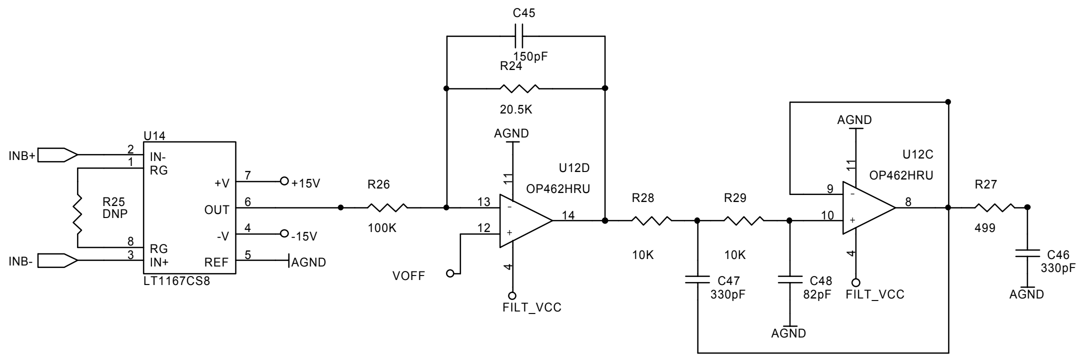


FIGURE 126. Servo16 Channel 11 Input Circuitry Schematic

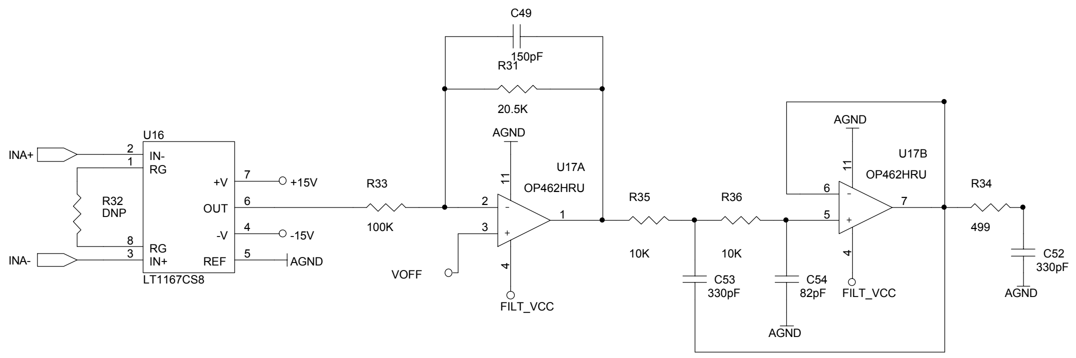


FIGURE 127. Servo16 Channel 12 Input Circuitry Schematic

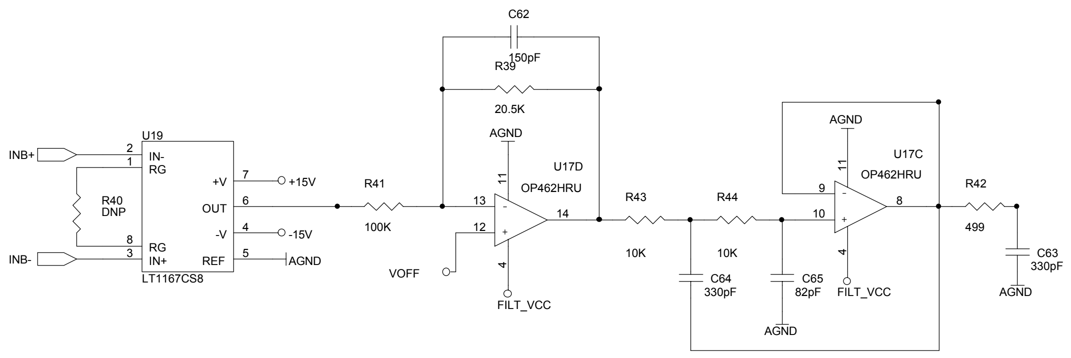


FIGURE 128. Servo16 Channel 13 Input Circuitry Schematic

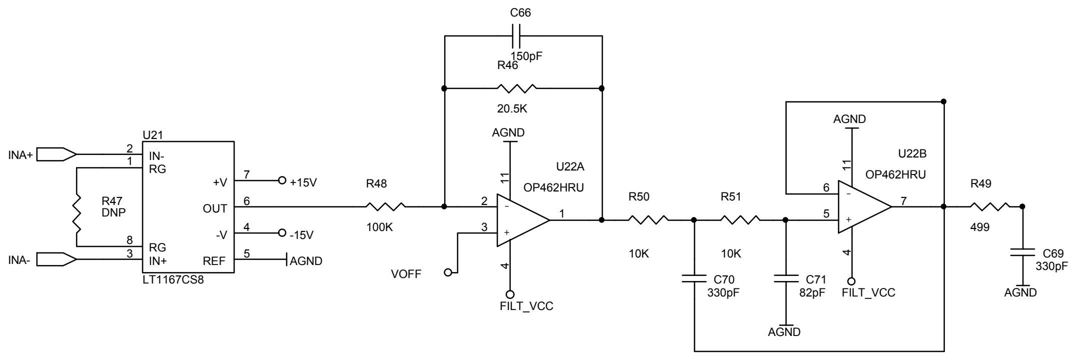


FIGURE 129. **Servo16 Channel 14 Input Circuitry Schematic**

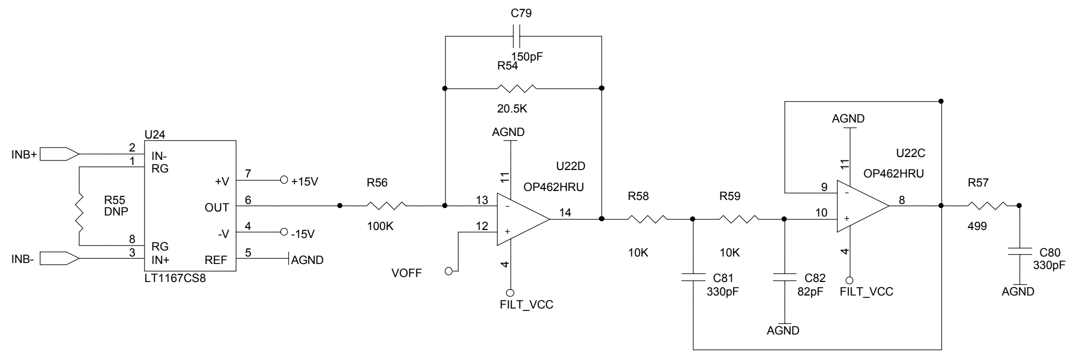


FIGURE 130. **Servo16 Channel 15 Input Circuitry Schematic**

Standard full-scale input range is +10 to -10V. Custom input ranges and filters are available for OEMs. Contact the sales department at Innovative Integration for more information.

The Servo16 A/D converter interface delivers two's complement 16-bit numbers for each channel as the data format. The following table gives example data values for the mid-scale, maximum, and minimum codes.

Input Voltage	A/D Output Code
+10V	0x7FFF
0 V	0
-10V	0x8000

The A/D used is the Texas Instruments (Burr-Brown) ADS8321. This is a low power, 16-bit A/D capable of sampling at up to 100 kHz.

A/D Triggering

The sample rate for the A/D is set by the DDS clock provided to the Servo16 from the host card (Chico-Plus, M6x etc). This clock must be 24 times the desired sample rate of the A/D. All A/D channels must run simultaneously from the same input clock. The Servo16 A/Ds may only be triggered to convert at the rate set by the sample clock. No provision is made for triggering based on an external signal.

A/D Error Compensation

The Servo16 provides a digital trim feature which allows incoming digital data samples from the A/D converter to be digitally trimmed for offset and gain errors before the data is stored to the FIFO buffer for retrieval by the OMNIBUS host. This trim is performed automatically in hardware with no software overhead (except initial setup).

Digital trim on each channel is based on a set of gain and offset trim coefficients stored to registers in the Servo16 logic. One pair of registers is provided per input channel to allow for independent trim of all 16 channels. The trim function is defined mathematically as follows:

Corrected data = (gain * A/D) + offset

Trim arithmetic is performed using fixed point hardware and results in a two's complement fixed point output. The gain coefficients are stored by the hardware as 17 bit numbers with a binary point between bits 15 and 16. This encoding results in a range of possible gain values from 0.0 to slightly below 2.0, where binary values with the MSB (bit 16) set result in gain values equal to or above 1.0 and binary values with the MSB clear result in gain values below 1.0. The following table gives example binary codes for several gain coefficients in order to illustrate the encoding method.

Binary Gain Coefficient Value	Gain Value
0x1C000	1.75
0x18000	1.5
0x10000	1.0
0x08000	0.5
0x04000	0.25
0x00000	0

TABLE 115. **Servo16 A/D Gain Coefficient Values**

Offset coefficients are stored as two's complement integers with a 16 bit width. Since the offset coefficients are signed, a negative offset coefficient value will cause the trim results to decrease in absolute value while a positive offset coefficient value will cause the results to increase in absolute value.

The gain and offset coefficients are stored in the gain and offset coefficient memory registers. 16 locations are provided for each type of coefficient.

Each module is delivered with a set of factory calibration coefficients that have been stored in the non-volatile on-module IDROM memory. Development system software for the OMNIBUS host card contains support for copying these coefficients from the IDROM to the coefficient registers.

Please note that digital trim does affect the absolute input range of the Servo16 since the trim process is performed in the digital domain (i.e. after conversion from the original analog signal, as opposed to a trim performed in the analog domain prior to conversion). Onboard offset and gain errors, although typically small and trimmable by the digital trim feature, will cause a digital dynamic range loss.

For example, in the case of an input channel which exhibited no gain error and an onboard offset error of +100 counts, the digital trim would typically be programmed to subtract 100 counts from the digitized data. This would have the effect of removing the offset error, but would result in a requirement that the input signal be limited to a range of approximately +/-9.970V in order to avoid exceeding the 16 bit digital dynamic range of the resulting trimmed data. The 30 mV of reduced amplitude is equal to 100 counts in the 16-bit digital realm, referenced to a nominal +/-10V input range (20 volts of input swing divided by 65536 counts of A/D converter dynamic range results in an analog bit resolution of approximately 300 microvolts). Since the onboard offset raises the input by 30 mV (100 counts of measured offset), the positive amplitude of +/-9.970 V input signal would just hit the maximum digital count value of the converter. But since the overall peak to peak swing of the signal is only 19.94 volts, approximately 200 counts of A/D dynamic range is unusable (i.e. the lowest converted value possible in this case is approximately -32568, instead of the theoretically lowest value of -32768).

IMPORTANT NOTE: the Servo16 digital trimming feature does not limit its output in the case of an over-range due to an excessively large absolute gain or offset coefficient or over-ranging due to the input signal. Digital outputs of the trim feature in such cases will swing past the normal resolution limit and wrap around through the opposite limit. For example, in the case of an input which is rising towards the positive digital rail, if the input continues to rise and the offset coefficient is set such that the resulting output data is larger than the maximum 16 bit amplitude, the data will wrap around and become a negative number with decreasing absolute amplitude.

Please note that this affect will not occur if the digital trim is set to a gain of 1.0 and an offset of zero (power-on reset defaults), due to the fact that the converter's output data cannot exceed the 16 bit 2's complement numerical range and the trim system is set to pass the data unchanged. The effect will also not occur if the gain coefficient is set to less than 1.0 and the offset is set to zero.

Users should pay careful attention to input signal absolute ranges and be sure that all coefficients are set to match the input signal. Users should also be careful that any changes to the instrumentation amplifier gain are also taken into account.

A/D FIFO

The A/D FIFO holds up to 256 sample pairs. Each sample pair is the error-corrected data from a pair of A/D converters. The ordering of the data is from lowest pair to highest pair of the pairs that are enabled by the A/D channel enable register (see below). For example, if pairs 0, 4 and 7 are enabled, the data will be put into the FIFO so that the first data read will be from channel pair 0, followed by pair 4 then pair 7. The next data will be channel 0 for the next sample period. Oldest data is always read first.

Each sample pair is stored in the FIFO with the even numbered channel of the pair in the least significant 16 bits and the odd numbered channel in the most significant 16 bits.

Even though during acquisition all channels are always running, each channel pair must be enabled before its data is transferred to the FIFO. This method of channel pair enabling allows the user to limit the data flow to just the channels required by the application, thus reducing data rate to the host processor and preserving FIFO storage.

A/D FIFO Threshold Register

The A/D FIFO threshold register allows software to control the minimum FIFO fullness level at which the A/D interrupt will be sent to the host. This feature allows the user to control the frequency and size of data movements to be performed, thus controlling the interrupt rate to the host processor. For servo applications, where minimum latency is desired, this threshold can be set to the number of pairs enabled. For data acquisition applications, this can be set to a larger size to reduce the interrupt rates to the host card.

Bit Field Name	Function
A/D FIFO Threshold 7 ..0	Programmable from 0 to 255 decimal. Default at reset is 255.

TABLE 116. **Servo16 A/D FIFO Threshold Register**

A/D Control Register

A/D reset state may be selected from the A/D control register. The following diagram gives the register definition.

Bit Number:	31-1	0
Bit Field:	Reserved	AD_RESET

FIGURE 131. **Servo16 A/D Control Register**

Bit Field Name	Function
AD_RESET	A/D FIFO and pipeline reset: 1=all A/Ds in reset, 0=all A/Ds out of reset.

TABLE 117. **Servo16 A/D Control Register Definition**

Asserting A/D reset clears the A/D buffer FIFOs and resets the calibration math hardware. Reset should be deasserted before enabling the AD_RUN control bit in the run control register.

A/D Channel Enable Register

The A/D channel enable register allows the user to select the channel pairs that will be collected into the FIFO. All pairs are disabled after reset. This register should only be written when the A/D data acquisition is stopped (see Run Control Register below).

Bit Number:	31-8	7	6	5	4	3	2	1	0
Bit Field:	Reserved	P7	P6	P5	P4	P3	P2	P1	P0

FIGURE 132. **Servo16 A/D Channel Enable Register**

Bit Field Name	Function
P0	1 = Pair 0 (ch 1/0) enabled, 0 = disabled
P1	1 = Pair 1 (ch 3/2) enabled, 0 = disabled
P2	1 = Pair 2 (ch 5/4) enabled, 0 = disabled
P3	1 = Pair 3 (ch 7/6) enabled, 0 = disabled
P4	1 = Pair 4 (ch 9/8) enabled, 0 = disabled
P5	1 = Pair 5 (ch 11/10) enabled, 0 = disabled
P6	1 = Pair 6 (ch 13/12) enabled, 0 = disabled
P7	1 = Pair 7 (ch 15/14) enabled, 0 = disabled

TABLE 118. **Servo16 A/D Channel Enable Register Definition**

D/A Output Circuitry

Each D/A channel of the Servo16 is an independent signal chain composed of a D/A and output analog filter. The D/A is Analog Devices AD5544 which is a 16-bit, 100 kHz update rate device. The D/A is a current output device which is translated to voltage in the successive stage then put through a simple one-pole reconstruction filter. The filter has a -3 dB corner frequency at 50 kHz. Standard output range is nominally -10 to +10 V full scale. The available output range may vary by as much as 1%. The following diagram shows the current to voltage conversion amp and analog filter.

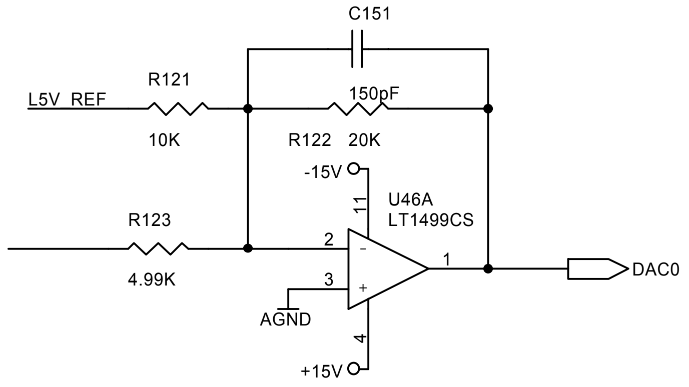


FIGURE 133. **Servo16 Channel 0 Output Circuitry Schematic**

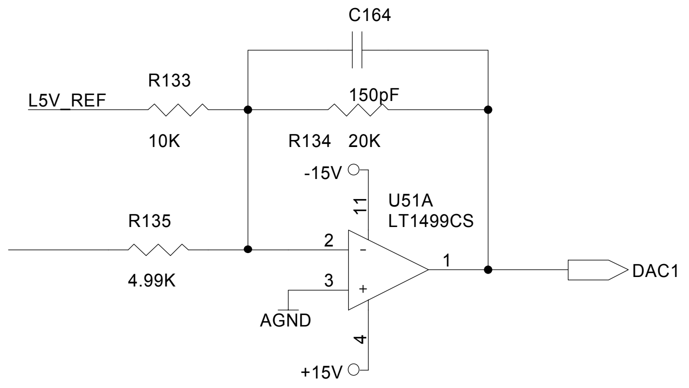


FIGURE 134. **Servo16 Channel 1 Output Circuitry Schematic**

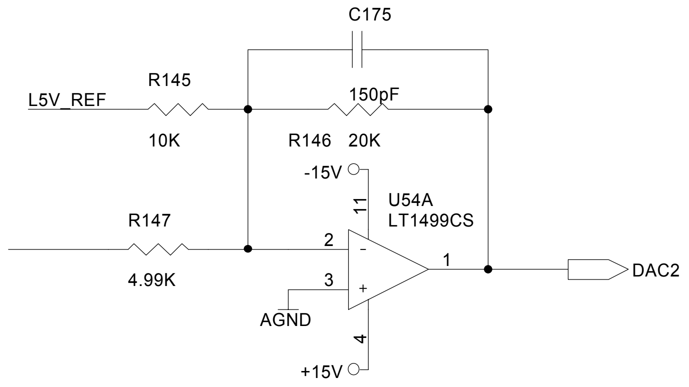


FIGURE 135. **Servo16 Channel 2 Output Circuitry Schematic**

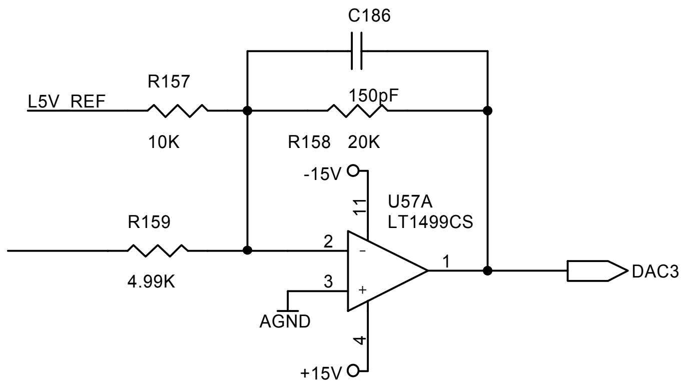


FIGURE 136. **Servo16 Channel 3 Output Circuitry Schematic**

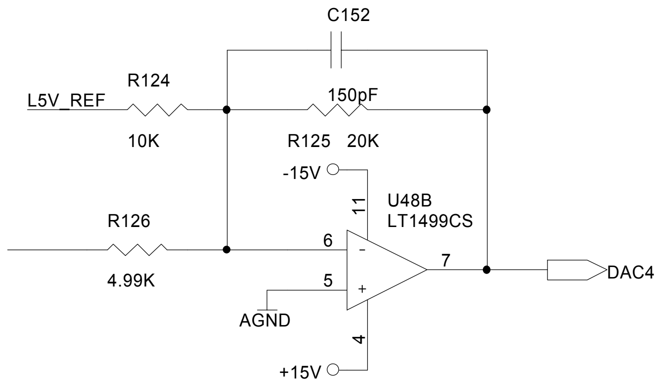


FIGURE 137. **Servo16 Channel 4 Output Circuitry Schematic**

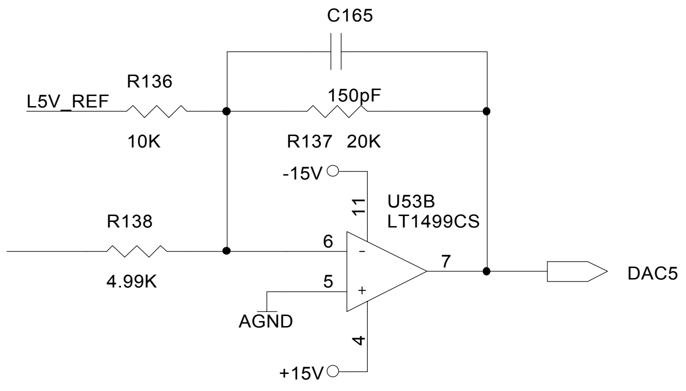


FIGURE 138. **Servo16 Channel 5 Output Circuitry Schematic**

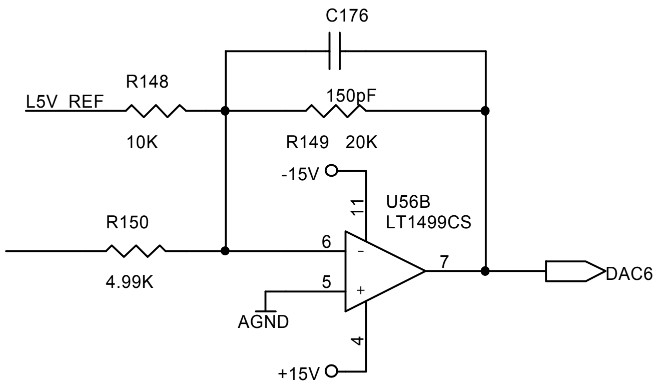


FIGURE 139. **Servo16 Channel 6 Output Circuitry Schematic**

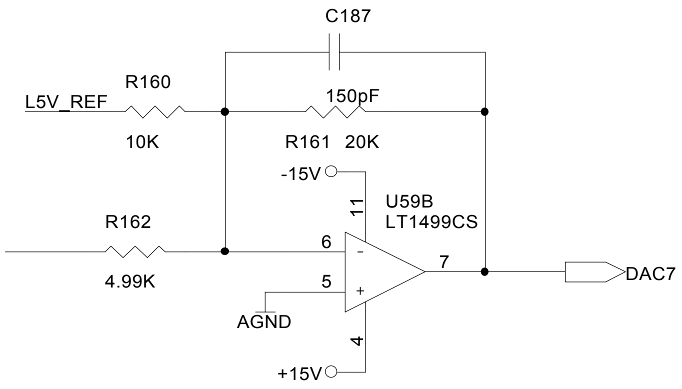


FIGURE 140. **Servo16 Channel 7 Output Circuitry Schematic**

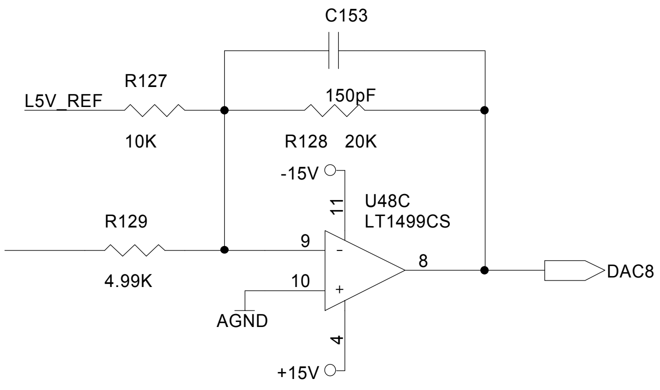


FIGURE 141. **Servo16 Channel 8 Output Circuitry Schematic**

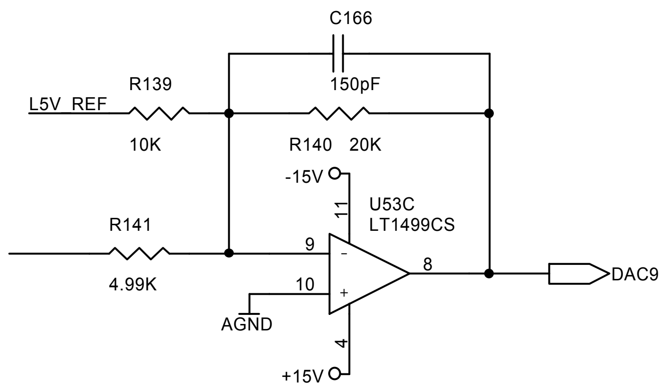


FIGURE 142. **Servo16 Channel 9 Output Circuitry Schematic**

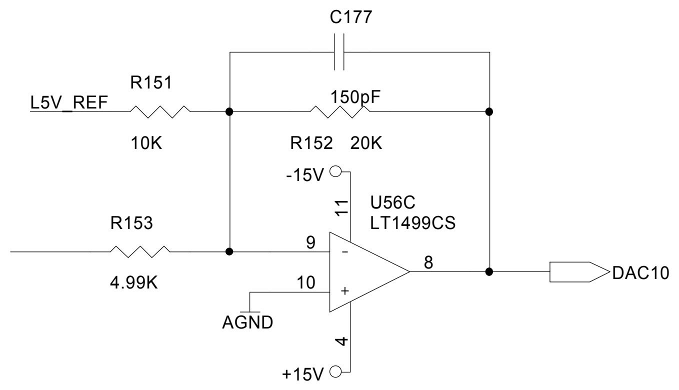


FIGURE 143. **Servo16 Channel 10 Output Circuitry Schematic**

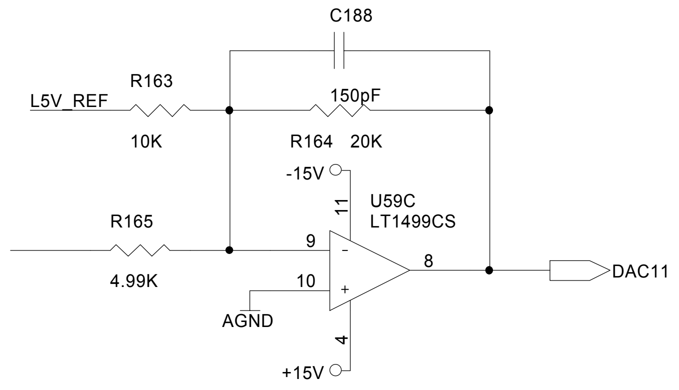


FIGURE 144. **Servo16 Channel 11 Output Circuitry Schematic**

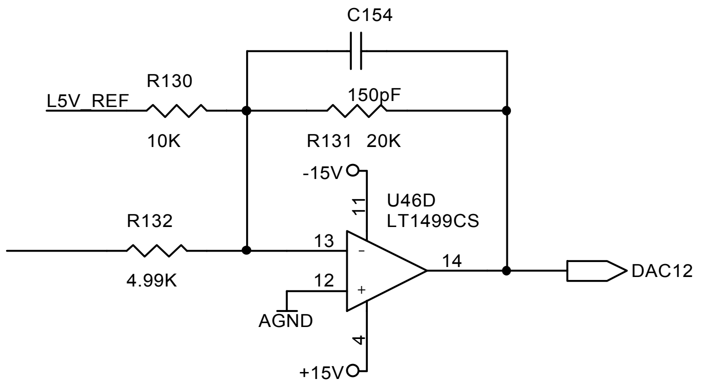


FIGURE 145. **Servo16 Channel 12 Output Circuitry Schematic**

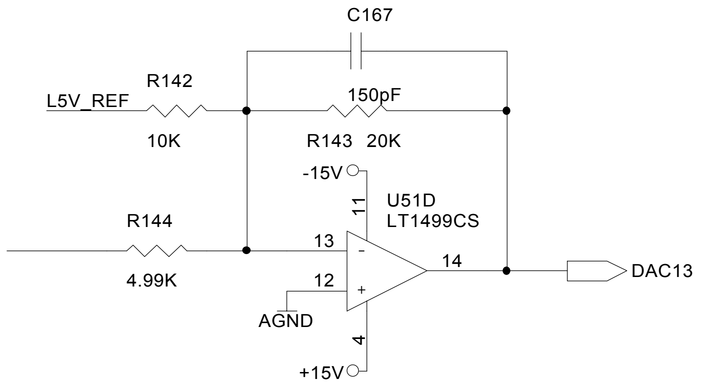


FIGURE 146. **Servo16 Channel 13 Output Circuitry Schematic**

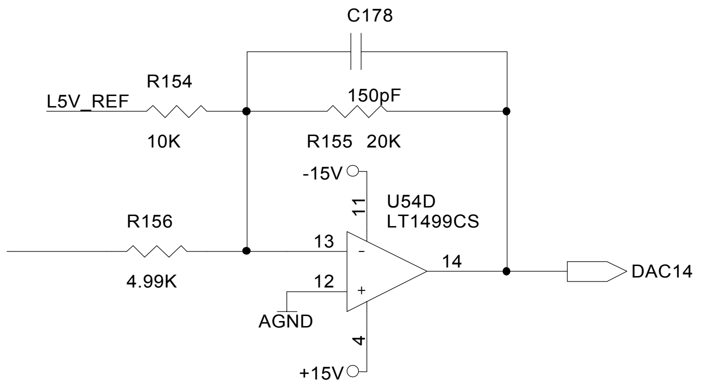


FIGURE 147. **Servo16 Channel 14 Output Circuitry Schematic**

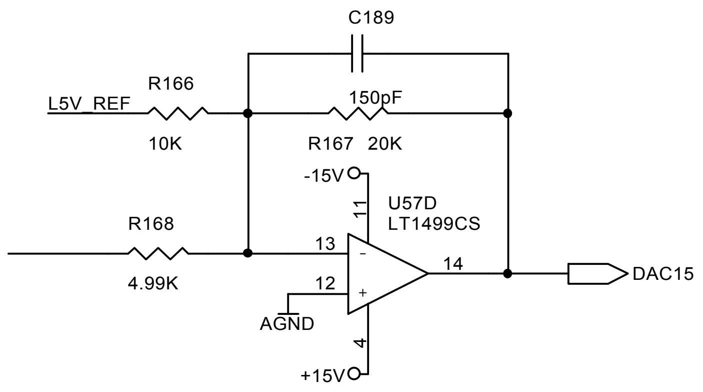


FIGURE 148. **Servo16 Channel 15 Output Circuitry Schematic**

Standard full-scale output range is +10 to -10V. Custom input ranges and filters are available for OEMs. Contact the sales department at Innovative Integration for more information.

The Servo16 D/A converter interface accepts two's complement 16-bit numbers for each channel as the data format. The following table gives example data values for the mid-scale, maximum, and minimum codes

Output Code	D/A Output Voltage
0x7FFF	+10V
0	0 V
0x8000	-10V

The update rate for the D/A is always synchronous with the A/D sample rate. All 16 channels are updated simultaneously at the same rate. The D/A update rate is equal one-24th of the input DDS clock rate to the module since this is required for the A/D. The clock is provided to the Servo16 from the host card (ChicoPlus, M6x etc).

The D/A has a serial data interface to the logic. The logic compensates the 16-bit output data for gain and offset errors and puts the data into a FIFO. At each D/A update period, the logic pulls the data from the FIFO and sends it to the D/A. This data is not converted to a voltage until the update trigger occurs so that the D/A voltages are all updated simultaneously at the precise update time.

D/A Error Correction

The Servo16 provides a digital trim feature which allows outgoing digital data samples to the D/A converter to be digitally trimmed for offset and gain errors before the data is written to the converters. This trim is performed automatically in hardware with no software overhead (except initial setup).

Digital trim on each channel is based on a set of gain and offset trim coefficients stored to registers in the Servo16 logic. One pair of registers is provided per input channel to allow for independent trim of all 16 channels. The trim function is defined mathematically as follows:

Corrected data = (gain * A/D) + offset

Trim arithmetic is performed using fixed point hardware and results in a two's complement fixed point output. The gain coefficients are stored by the hardware as 17 bit numbers with a binary point between bits 15 and 16. This encoding results in a range of possible gain values from 0.0 to slightly below 2.0, where binary values with the MSB (bit 16) set result in gain values equal to or above 1.0 and binary values with the MSB clear result in gain values below 1.0. The following table gives example binary codes for several gain coefficients in order to illustrate the encoding method.

Binary Gain Coefficient Value	Gain Value
0x1C000	1.75
0x18000	1.5
0x10000	1.0
0x08000	0.5
0x04000	0.25
0x00000	0

TABLE 119. **Servo16 D/A Gain Coefficient Values**

Offset coefficients are stored as 16-bit fixed point numbers with a zero point at the value 0x8000. The coefficient value required to obtain a particular offset may be calculated as follows:

Coefficient = offset + 0x8000

Hence an offset of zero is set by using a coefficient of 0x8000, an offset of +1 by using a coefficient of 0x8001, and an offset of -1 by using a coefficient of 0x7FFF.

The gain and offset coefficients are stored in the gain and offset coefficient memory registers. 16 locations are provided for each type of coefficient.

Each module is delivered with a set of factory calibration coefficients that have been stored in the non-volatile on-module IDROM memory. Development system software for the OMNIBUS host card contains support for copying these coefficients from the IDROM to the coefficient registers.

Please note that digital trim does affect the absolute input range of the Servo16 since the trim process is performed in the digital domain (i.e. after conversion from the original analog signal, as opposed to a trim performed in the analog domain prior to conversion). Onboard offset and gain errors, although typically small and trimmable by the digital trim feature, will cause a digital dynamic range loss.

For example, in the case of an input channel which exhibited no gain error and an onboard offset error of +100 counts, the digital trim would typically be programmed to subtract 100 counts from the digitized data. This would have the effect of removing the offset error, but would result in a requirement that the input signal be limited to a range of approximately +/-9.970V in order to avoid exceeding the 16 bit digital dynamic range of the resulting trimmed data. The 30 mV of reduced amplitude is equal to 100 counts in the 16-bit digital realm, referenced to a nominal +/-10V input range (20 volts of input swing divided by 65536 counts of A/D converter dynamic range results in an analog bit resolution of approximately 300 microvolts). Since the onboard offset raises the input by 30 mV (100 counts of measured offset), the positive amplitude of +/-9.970 V input signal would just hit the maximum digital count value of the converter. But since the overall peak to peak swing of the signal is only 19.94 volts, approximately 200 counts of A/D dynamic range is unusable (i.e. the lowest converted value possible in this case is approximately -32568, instead of the theoretically lowest value of -32768).

IMPORTANT NOTE: The Servo16 digital trimming feature does not limit its output in the case of an over-range due to an excessively large absolute gain or offset coefficient or over-ranging due to the input signal. Digital outputs of the trim feature in such cases will swing past the normal resolution limit and wrap around through the opposite limit. For example, in the case of an input which is rising towards the positive digital rail, if the input continues to rise and the offset coefficient is set such that the resulting output data is larger than the maximum 16 bit amplitude, the data will wrap around and become a negative number with decreasing absolute amplitude.

Please note that this affect will not occur if the digital trim is set to a gain of 1.0 and an offset of zero (power-on reset defaults), due to the fact that the converter's output data cannot exceed the 16 bit 2's complement numerical range and the trim system is set to pass the data unchanged. The effect will also not occur if the gain coefficient is set to less than 1.0 and the offset is set to zero.

Users should pay careful attention to input signal absolute ranges and be sure that all coefficients are set to match the input signal. Users should also be careful that any changes to the instrumentation amplifier gain are also taken into account

D/A Data FIFO

The D/A FIFO holds up to 256 sample pairs. Each sample pair is the error-corrected data from the OMNIBUS host. The ordering of the data is from lowest pair to highest pair of the pairs that are enabled by the A/D channel enable register (see below). For example, if pairs 0, 4 and 7 are enabled, the data should be written to the FIFO from the host so that the first data read will be from channel pair 0, followed by pair 4 then pair 7. The next data will be channel 0 for the next sample period.

Each sample pair is stored in the FIFO with the even numbered channel of the pair in the least significant 16 bits and the odd numbered channel in the most significant 16 bits.

Even though during output all channels are always running, each channel pair must be enabled before its data is transferred to the FIFO. This method of channel pair enabling allows the user to limit the data flow to just the channels required by the application, thus reducing data rate to the host processor and preserving FIFO storage. Disabled channels have the mid-scale data value supplied to them by onboard logic (nominally resulting in a zero volt output).

D/A FIFO Threshold Register

The D/A FIFO threshold register allows software to control the maximum level of available FIFO space at which the D/A interrupt will be sent to the host. An interrupt from the D/A FIFO is asserted when the amount of space available in the FIFO above the programmable threshold level. This allows the user to control the frequency and size of data movements to be performed, thus controlling the interrupt rate to the host processor. For servo applications, where minimum latency is desired, this threshold can be set to the number of pairs enabled. For data acquisition applications, this can be set to a larger size to reduce the interrupt rates to the host card.

Bit Field Name	**Function**
D/A FIFO Threshold 7 ..0	Programmable from 0 to 255 decimal. Default at reset is 255.

TABLE 120. **Servo16 D/A FIFO Threshold Register**

D/A Control Register

The D/A reset state and conversion trigger signal source may be selected from the D/A control register. The following diagram gives the register definition.

Bit Number:	31-1	0
Bit Field:	Reserved	DA_RESET

FIGURE 149. **Servo16 D/A Control Register**

Bit Field Name	**Function**
DA_RESET	D/A FIFO and pipeline reset: 1=all D/As in reset, 0=all D/As out of reset.

TABLE 121. **Servo16 D/A Control Register Definition**

Asserting D/A reset clears the D/A buffer FIFOs and resets the calibration math hardware. Reset should be deasserted before enabling the DA_RUN control bit in the run control register.

D/A Channel Enable Register

The D/A channel enable register allows the user to select the channel pairs that will be collected into the FIFO. All pairs are disabled after reset. This register should only be written when D/A signal generation is stopped (see Run Control Register below).

Bit Number:	31-8	7	6	5	4	3	2	1	0
Bit Field:	Reserved	P7	P6	P5	P4	P3	P2	P1	P0

FIGURE 150. **Servo16 D/A Channel Enable Register**

Bit Field Name	Function
P0	1 = Pair 0 (ch 1/0) enabled, 0 = disabled
P1	1 = Pair 1 (ch 3/2) enabled, 0 = disabled
P2	1 = Pair 2 (ch 5/4) enabled, 0 = disabled
P3	1 = Pair 3 (ch 7/6) enabled, 0 = disabled
P4	1 = Pair 4 (ch 9/8) enabled, 0 = disabled
P5	1 = Pair 5 (ch 11/10) enabled, 0 = disabled
P6	1 = Pair 6 (ch 13/12) enabled, 0 = disabled
P7	1 = Pair 7 (ch 15/14) enabled, 0 = disabled

TABLE 122. **Servo16 D/A Channel Enable Register Definition**

Run Control Register

The run control register enables data movement between the converters, DSP logic, and FIFOs. Separate controls are provided for each of the I/O directions, allowing the A/D and D/A converters to be enabled independently. Both directions may be simultaneously enabled by setting both bits of the control register active using a single write to the register.

Bit Number:	31-2	1	0
Bit Field:	Reserved	DAC_RUN	AD_RUN

FIGURE 151. **Servo16 Run Control Register**

Bit Field Name	Function
DAC_RUN	Run/Stop D/A: 1=run, 0=stop
AD_RUN	Run/Stop A/D: 1=run, 0=stop

TABLE 123. **Servo16 Run Control Register Definition**

Servo16 Calibration

The Servo16 is calibrated at the factory prior to delivery. A test report delivered with each module shows the results of the calibration. Normally, the module should remain in calibration for a very long time since the errors due to analog trim adjustments have been eliminated with digital error correction methods. However, the module may occasionally need calibration over time or if the operating temperature is significantly different than 25 degrees Celsius.

If the module requires calibration, you can use the example programs to update the calibration coefficients. Required equipment is a calibrated 5Vreference voltage, accurate to 300 uV, with low noise. Each channel is first measured with ground connected, then with 5V reference connected. The software calculates the gain and offset error coefficients and updates the calibration memory on the module.

You may also return the module to Innovative Integration for calibration if desired. Contact the sales department for this service.

Initialization Issues

The following steps are required for proper initialization and operation of the converters on the Servo16. Normally, the initialization software from Innovative Integration in the Zuma Toolkit or Chico Armada system performs these functions. If you plan to write your own routines for custom applications, here are the required initialization steps.

1. Set the calibration coefficients for the A/D and D/A converters.
1. Initialize the host DDS timebase signal to 24 times the required sample rate.
2. Toggle the converter reset controls active, then inactive.
3. Deassert the run control bits (stop the converters).
4. Set the converter channel enable registers.
5. Set the FIFO threshold registers.
6. Initialize and enable the required host interrupt hardware. On hosts which support it, interrupts should be configured to active low, level sensitive inputs.
7. Set the required run bits.

In applications which use only the D/A converters, the A/D reset bit must be set inactive even though A/D data is not being acquired. This is due to the Servo16's use of internal A/D timing to control the D/A sample rate.

Firmware Updating for the Servo16

Should the Servo16 firmware ever need updating for a bug fix or feature improvement, the firmware may be updated by the host card. The FPGA logic firmware is held in a re-programmable memory that can be rewritten using an update program included in the support software from Innovative Integration. Firmware updates are only possible on modules that are communicating properly with the host and should not be attempted on damaged modules.

Up to date support firmware is available from Innovative Integration web site. Contact the support staff at Innovative Integration if you need assistance updating your module.

Servo16 Heat Management

The Servo16 module dissipates significant amounts of heat during use, and requires a minimum of 35 CFM of forced air cooling for reliable operation.